FINAL REPORT

DUCT FLOW NONUNIFORMITIES FOR SPACE SHUTTLE MAIN ENGINE (SSME)

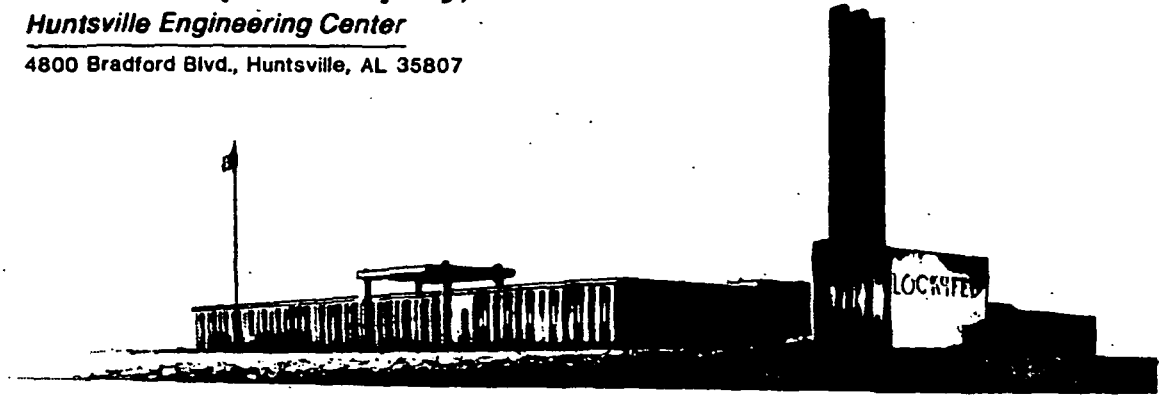
30 December 1987

Contract NAS8-35592

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
MARSHALL SPACE FLIGHT CENTER, AL 35812

By

Lockheed
Missiles & Space Company, Inc.
Huntsville Engineering Center
4800 Bradford Blvd., Huntsville, AL 35807



FOREWORD

This report was prepared by personnel of the Computational Mechanics Section of Lockheed's Huntsville Engineering Center. It constitutes final documentation of efforts performed under Contract NAS8-35592 for NASA-Marshall Space Flight Center.

The NASA-MSFC Contracting Officer's Representative for this research study was Dr. P.K. McConnaughey, ED32.

CONTENTS

1. INTRODUCTION

Future research development, as well as production activities in space by U.S. federal and private agencies, will depend on the Space Shuttle and its derivative versions as a principal space transportation system. This dependence requires improved designs or techniques to extend the life, upgrade performance, reduce weight, lower operational costs, and generally improve the functional capability of the main propulsion system. The engines for this main propulsion system are advanced high pressure engines operating on oxygen and hydrogen. A need therefore exists to investigate, develop, and define basic concepts in support of the main propulsion system improvements. One basic area that has been investigated is the hot gas flow nonuniformities that occur within the manifold, duct work, and main injector. Nonuniformities result from highly distorted and mismatched flows within the ducts which create severe environments for the system components, thus limiting their useful life.

Development and verification tests of the Space Shuttle Main Engine (SSME) and results of computational modeling have shown that the three gas transfer tubes have an uneven flow distribution with large areas of separated flow. The outer transfer tubes each carry approximately twice the amount of gas as the center tube. This causes the energy of the gas to be much higher in the outer tubes. Flow from the tubes impinge upon the main injector liquid oxygen posts which bend under the static load of the gas flow, the bending being more pronounced in line with the outer tubes. To alleviate this phenomenon and to keep the posts cooler, shields, linking pairs of posts in the outermost row, were installed. The design alteration enhanced the injector life, however, LOX post failures show that this change alone is insufficient to grant specified life at equal to or greater rated power levels. Incorporation of the shields also affects circumferential flow in the annulus and degrades the engine performance and necessitates higher operating temperatures in the

turbines. To improve the SSME design and for future use in the design of new generation rocket engines, more experimental data are needed. These data, combined with previous test data and integrated into analytical methods, can be used to establish a base for design of high energy hot gas flow systems.

In addition the preburners of the SSME still experience serious problems related also to the cracking of LOX posts and the frisbee in the inlet manifolds. Operation of the fuel preburner creates flows which cause turbine blade cracking in the high pressure fuel turbopump. The most likely cause of these problems is the extreme thermal environments caused by start-up and shut-down of the engine. A detailed transient analysis of these flows is necessary in order to help alleviate these problems.

This report describes results of efforts by personnel of the Computational Mechanics Group at the Lockheed-Huntsville Engineering Center to assist the computational staff of NASA-MSFC in developing analytical capabilities for modeling hot gas flow on the fuel side of the SSME. The primary objective was to develop and deliver a computer code which produces a computational grid for the three-duct Hot Gas Manifold (HGM) on the fuel side of the SSME. A secondary objective was to provide input for an accurate computation of the flow characteristics inside the fuel side preburner during the start-up transient of the SSME.

2. TECHNICAL APPROACH

2.1 HGM COMPUTATIONAL GRID CODE

Nearly all computational codes which are available to numerically solve the three-dimensional fluid flow equations are designed to be applied to a well structured grid model of the flow region. To perform the calculations, generalized independent variables are introduced which transform the physical coordinates, (x, y, z), into general curvilinear coordinates, (η_1, η_2, η_3). Thus the physical domain must be gridded as a single or series of hexahedral zones described by eight corner points, 12 edges, and six surfaces. Such an arbitrary zone is shown in Fig. 2-1.

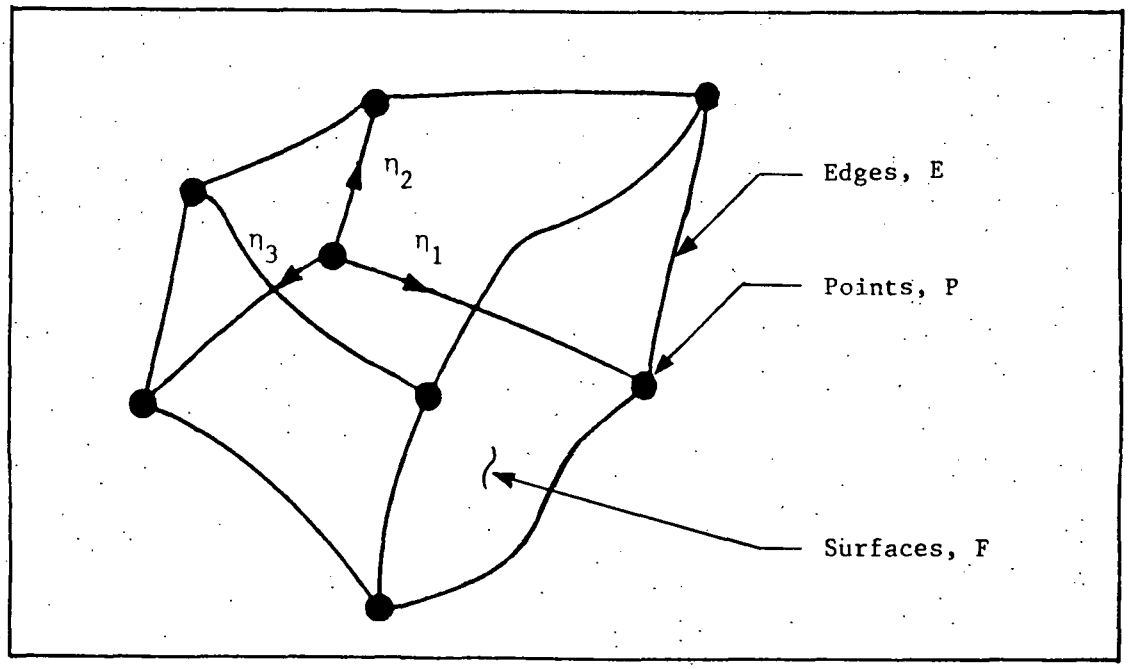


Fig. 2-1 Hexahedral Element Showing Local Intrinsic Coordinates

The approach used in the current study was to provide an algebraic grid generation code which would produce the Cartesian coordinates (x, y, z), for points along the lines of constant η_1, η_2, and η_3. Basic mathematical techniques taken from analytic geometry and vector algebra were employed to describe a hexahedral zone in terms of piecewise continuous analytic functions which represent the zonal edges and surfaces.

An intrinsic curvilinear coordinate system can be produced by mapping a unit cube onto the simply connected hexahedral zone. What is needed is a transformation function that will map a unit cube in (η_1, η_2, η_3) space univalently onto the hexahedral volume of interest thus producing the required intrinsic coordinate system. A procedure which produces the desired result is referred to as either the method of transfinite interpolation or multi-variate blending function interpolation (Ref. 1). A brief description of this method follows.

Let $F(\eta_1, \eta_2, \eta_3)$ be a vector-valued functional representing the region R of interest in curvilinear space. Then as (η_1, η_2, η_3) range over R, F traces out the region in Euclidean space (x, y, z). Also let ϕ, Ψ, and λ be blending functions which obey the cardinality conditions:

$$\phi_i(\eta_1=l) = \begin{cases} 1 \text{ if } i = 1 \\ 0 \text{ if } i \neq 1 \end{cases}$$

$$\psi_j(\eta_2=m) = \begin{cases} 1 \text{ if } j = m \\ 0 \text{ if } j \neq m \end{cases}$$

$$\lambda_k(\eta_3=n) = \begin{cases} 1 \text{ if } k = n \\ 0 \text{ if } k \neq n \end{cases}$$

The simplest form of blending functions meeting these conditions are

$$\phi_o(\eta_1) = 1 - \eta_1 \qquad \phi_1(\eta_1) = \eta_1$$
$$\psi_o(\eta_2) = 1 - \eta_2 \qquad \psi_1(\eta_2) = \eta_2$$
$$\lambda_o(\eta_3) = 1 - \eta_3 \qquad \lambda_1(\eta_3) = \eta_3$$

Then a trilinearly blended interpolant of F, which will map a unit cube onto the region R is given by

$$U(\eta_1,\eta_2,\eta_3) = \begin{bmatrix} x(\eta_1,\eta_2,\eta_3) \\ y(\eta_1,\eta_2,\eta_3) \\ z(\eta_1,\eta_2,\eta_3) \end{bmatrix}$$

$$= (1-\eta_1)F(0\ \eta_2,\eta_3) + \eta_1 F(1,\eta_2,\eta_3)$$

$$+ (1-\eta_2)F(\eta_1,0,\eta_3) + \eta_2 F(\eta_1,1,\eta_3)$$

$$+ (1-\eta_3)F(\eta_1,\eta_2,0) + \eta_3 F(\eta_1,\eta_2,1)$$

$$- (1-\eta_1)(1-\eta_2))F(0,0,\eta_3) - (1-\eta_1)\eta_2 F(0,1,\eta_3)$$

$$- \eta_1(1-\eta_2)F(1,0,\eta_3) - \eta_1\eta_2 F(1,1,\eta_3)$$

$$- (1-\eta_1)\eta_3 F(0,\eta_2,0) - (1-\eta_1)\eta_3 F(0,\eta_2,1)$$

$$- \eta_1(1-\eta_3)F(1,\eta_2,0) - \eta_1\eta_3 F(1,\eta_2,1)$$

$$- (1-\eta_2)(1-\eta_3)F(\eta_1,0,0) - (1-\eta_2)\eta_3 F(\eta_1 0,1)$$

$$- \eta_2(1-\eta_3)F(\eta_1,1,0) - \eta_2\eta_3 F(\eta_1,1,1)$$

$$+ (1-\eta_1)(1-\eta_2)(1-\eta_3)F(0,0,0) + (1-\eta_1)(1-\eta_2)\eta_3 F(0,0,1)$$

$$+ (1-\eta_1)\eta_2(1-\eta_3)F(0,1,0) + (1-\eta_1)\eta_2\eta_3 F(0,1,1)$$

$$+ \eta_1(1-\eta_2)(1-\eta_3)F(1,0,0) + \eta_1(1-\eta_2)\eta_3 F(1,0,1)$$

$$+ \eta_1\eta_2(1-\eta_3)F(1,1,0) + \eta_1\eta_2\eta_3 F(1,1,1)$$

where

$F(0,\eta_2,\eta_3)$ represents a surface with $\eta_1=0$, etc.
$F(0,0,\eta_3)$ represents an edge with $\eta_1=\eta_2=0$, etc.
$F(0,0,0)$ represents a point with $\eta_1=\eta_2=\eta_3=0$, etc.

In two dimensions this equation performs a bilinear interpolation over an arbitrary region consisting of four distinct corners simply connected by four edges, where

$$F(0,\eta_2) = \begin{bmatrix} x \\ y \end{bmatrix} \text{ along } EDGE_4 \qquad F(0,0) = \begin{bmatrix} x \\ y \end{bmatrix} \text{ at } POINT_1$$

$$F(1,\eta_2) = \begin{bmatrix} x \\ y \end{bmatrix} \text{ along } EDGE_2 \qquad F(0,1) = \begin{bmatrix} x \\ y \end{bmatrix} \text{ at } POINT_4$$

$$F(\eta_1,0) = \begin{bmatrix} x \\ y \end{bmatrix} \text{ along } EDGE_1 \qquad F(1,0) = \begin{bmatrix} x \\ y \end{bmatrix} \text{ at } POINT_2$$

$$F(\eta_1,1) = \begin{bmatrix} x \\ y \end{bmatrix} \text{ along } EDGE_3 \qquad F(1,1) = \begin{bmatrix} x \\ y \end{bmatrix} \text{ at } POINT_3$$

and the interpolation equation for F could be rewritten as

$$U = \begin{bmatrix} x \\ y \end{bmatrix} = (1-\eta_1)EDGE_4 + \eta_1 EDGE_2 + (1-\eta_2)EDGE_1 + \eta_2 EDGE_3$$
$$- (1-\eta_1)(1-\eta_2)POINT_1 - (1-\eta_1)\eta_2 POINT_4$$
$$- \eta_1(1-\eta_2)POINT_2 - \eta_1\eta_2 POINT_3$$

In this equation $EDGE_4$ represents a vector-valued functional along edge four, etc. Examination of this equation shows that it performs linear interpolations between $EDGE_1$ and $EDGE_3$ and between $EDGE_2$ and $EDGE_4$, hence the term bilinear interpolation. Hence, if the functional for each edge can be derived and is analytic, a grid or mesh of node points can be generated by substituting values of η_1 and η_2 into the equation.

In three dimensions the general equation above performs trilinearly blended interpolation over an arbitrary region consisting of eight distinct corner points simply connected by 12 edges, where

$$F(0,\eta_2,\eta_3) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ on } SIDE_5, \text{ etc.}$$

$$F(0,0,\eta_3) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ along } EDGE_5, \text{ etc.}$$

$$F(0,0,0) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ at } POINT_1, \text{ etc.}$$

and which can be rewritten as

$$\begin{aligned} U = \begin{bmatrix} x \\ y \\ z \end{bmatrix} &= (1-\eta_1)SIDE_5 + \eta_1 SIDE_6 + (1-\eta_2)SIDE_2 + \eta_2 SIDE_4 + (1-\eta_3)SIDE_1 + \eta_3 SIDE_3 \\ &- (1-\eta_1)(1-\eta_2)EDGE_5 - (1-\eta_1)\eta_2 EDGE_8 - \eta_1(1-\eta_2)EDGE_6 - \eta_1\eta_2 EDGE_7 \\ &- (1-\eta_1)(1-\eta_3)EDGE_4 - (1-\eta_1)\eta_3 EDGE_{12} - \eta_1(1-\eta_3)EDGE_2 - \eta_1\eta_3 EDGE_{10} \\ &- (1-\eta_1)(1-\eta_3)EDGE_1 - (1-\eta_2)\eta_3 EDGE_9 - \eta_2(1-\eta_3)EDGE_3 - \eta_2\eta_3 EDGE_{11} \\ &+ (1-\eta_1)(1-\eta_2)(1-\eta_3)POINT_1 + (1-\eta_1)(1-\eta_2)\eta_3 POINT_5 + (1-\eta_1)\eta_2(1-\eta_3)POINT_4 \\ &+ (1-\eta_1)\eta_2\eta_3 POINT_8 + \eta_1(1-\eta_2)(1-\eta_3)POINT_2 + \eta_1(1-\eta_2)\eta_3 POINT_6) \\ &+ \eta_1\eta_2(1-\eta_3)POINT_3 + \eta_1\eta_2\eta_3 POINT_7 \end{aligned}$$

where $SIDE_i$, $EDGE_j$, $POINT_k$ represent vector-valued functionals on the surfaces, along the edges, and at the corner points, respectively.

This equation reduces to the previous two-dimensional analog along any flat surface or along any surface in which a straight line can be drawn between any two opposing edges such that the line lies entirely within the surface.

With the general transformation, any point in local coordinates η_1, η_2, η_3 can be related to the physical Cartesian coordinates x, y, z. The entire grid of discrete points is generated in the HGM code using this concept. This general interpolant can accommodate any stretching function for concentrating points near walls or regions of large gradients. Furthermore, the edges of the hexahedral can be segmented allowing another means of grid spacing control.

2.2 SSME FUEL-SIDE PREBURNER ANALYSIS

Operation of the fuel preburner is initiated by opening propellant control valves and igniting the flow with the augmented spark ignited (ASI). The configuration of the FPB is shown in Fig. 2-2. The flow passage from the fuel preburner oxidizer valve (FPOV) into the oxidizer manifold above the interpropellant plate is shown in this figure. The oxidizer flow to the ASI is supplied from a bleed line from the FPOV. The valves which control this flow are shown schematically in Fig. 2-3; note that there is not a separate fuel control valve for each preburner. Flows from the fuel and oxidizer manifolds enter the FPB combustor through the injector face plate, one third of the layout of which is shown in Fig. 2-2. The preburner liner extension shown in Fig. 2-2 slides inside of the sealer groove as shown in the upper left and right sides of Fig. 2-5. Thus, the preburned gases flow, via the dome at the bottom of the chamber, onto the turbine blades, then following the path through the 180-deg turnaround duct to the fuel bowl and transfer ducts of the HGM and onto the LOX posts and into the main combustion chamber.

The presence of the baffles shown in Figs. 2-2 and 2-4 create symmetry planes for computational analysis but remove the simplicity of axis symmetry. In addition, any model geometry should include the influence of the large nut at the bottom of the preburner chamber which holds the dome in place.

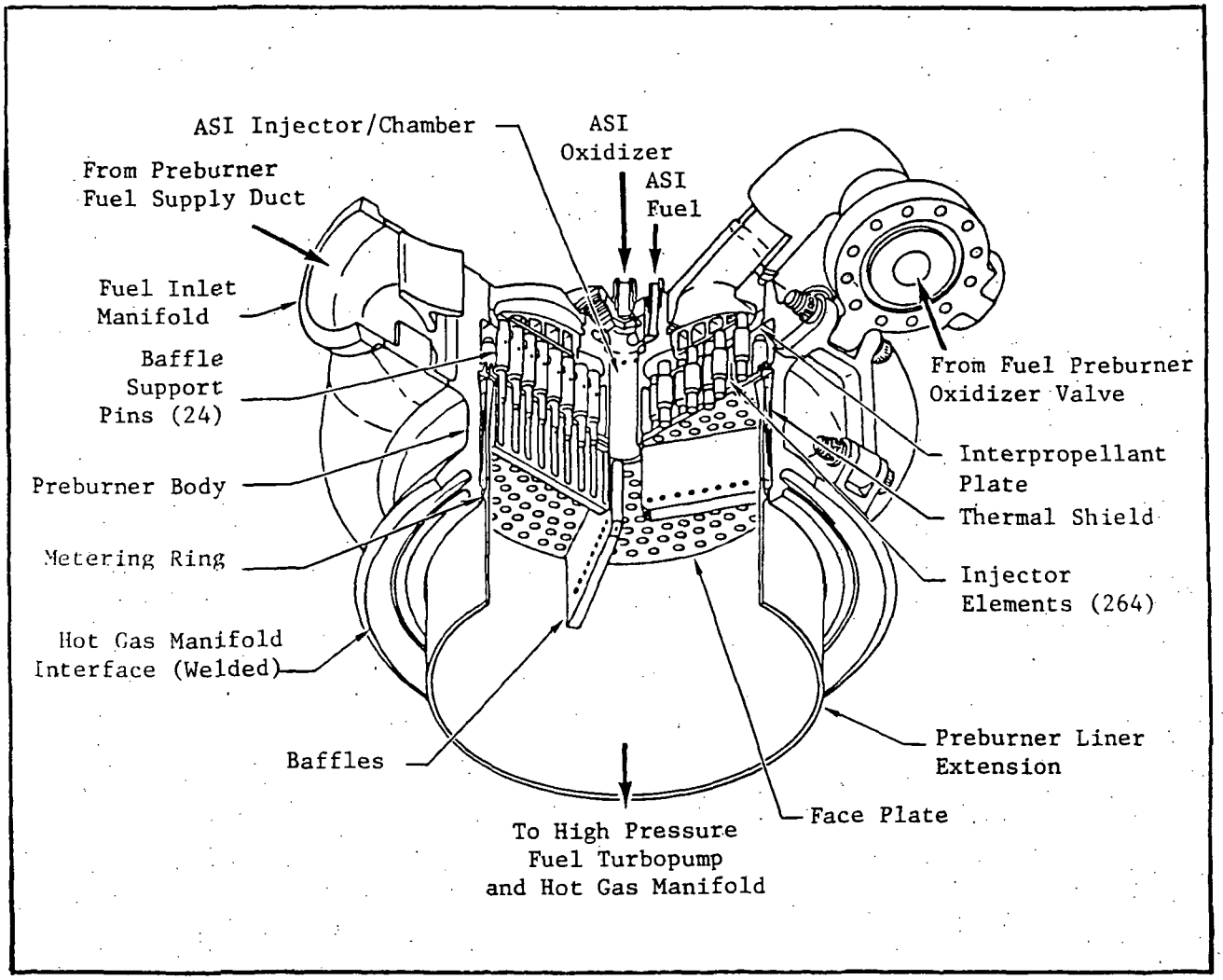


Fig. 2-2 Fuel Preburner Configuration

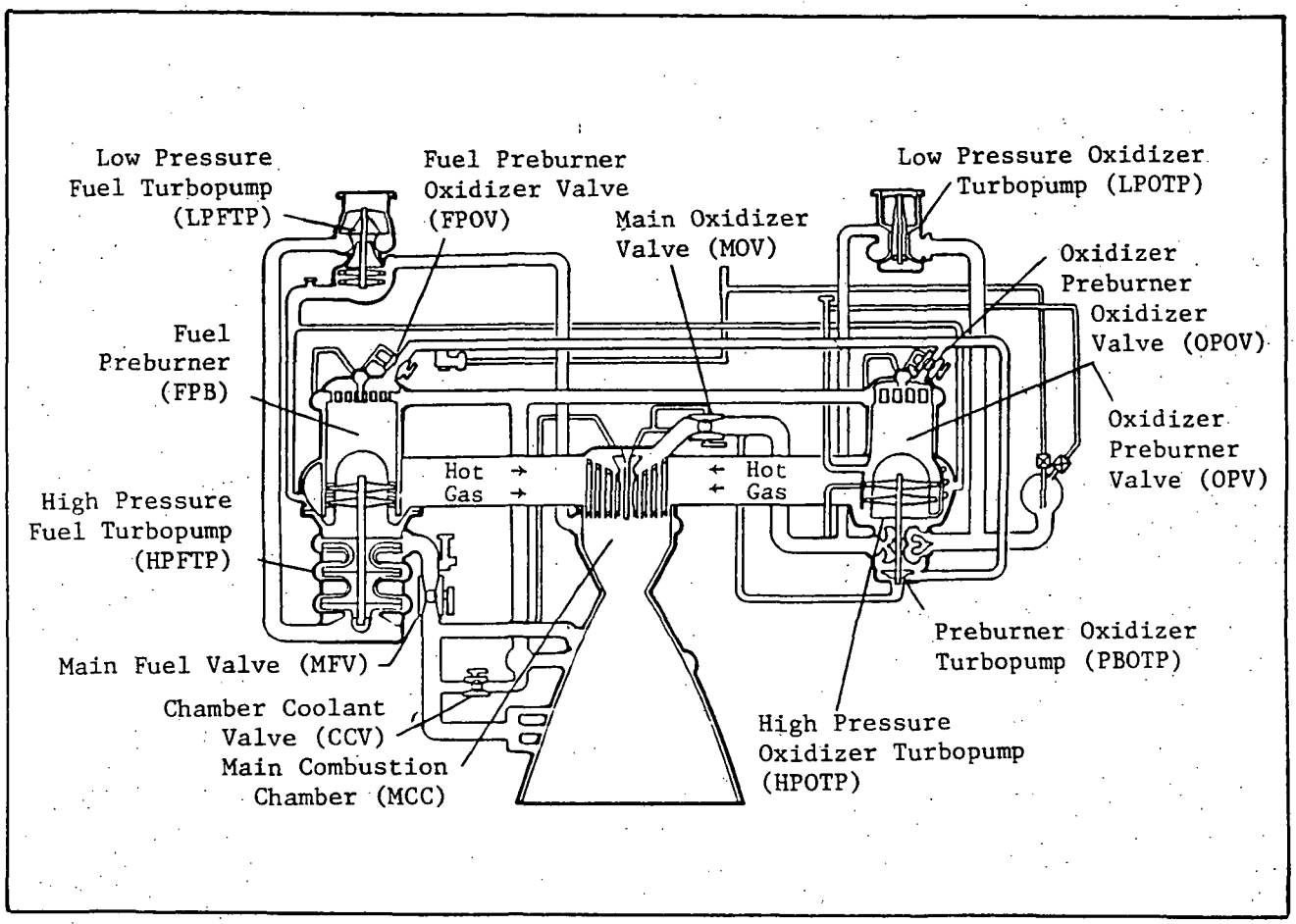


Fig. 2-3 Schematic of Control Valves for Preburner Configuration

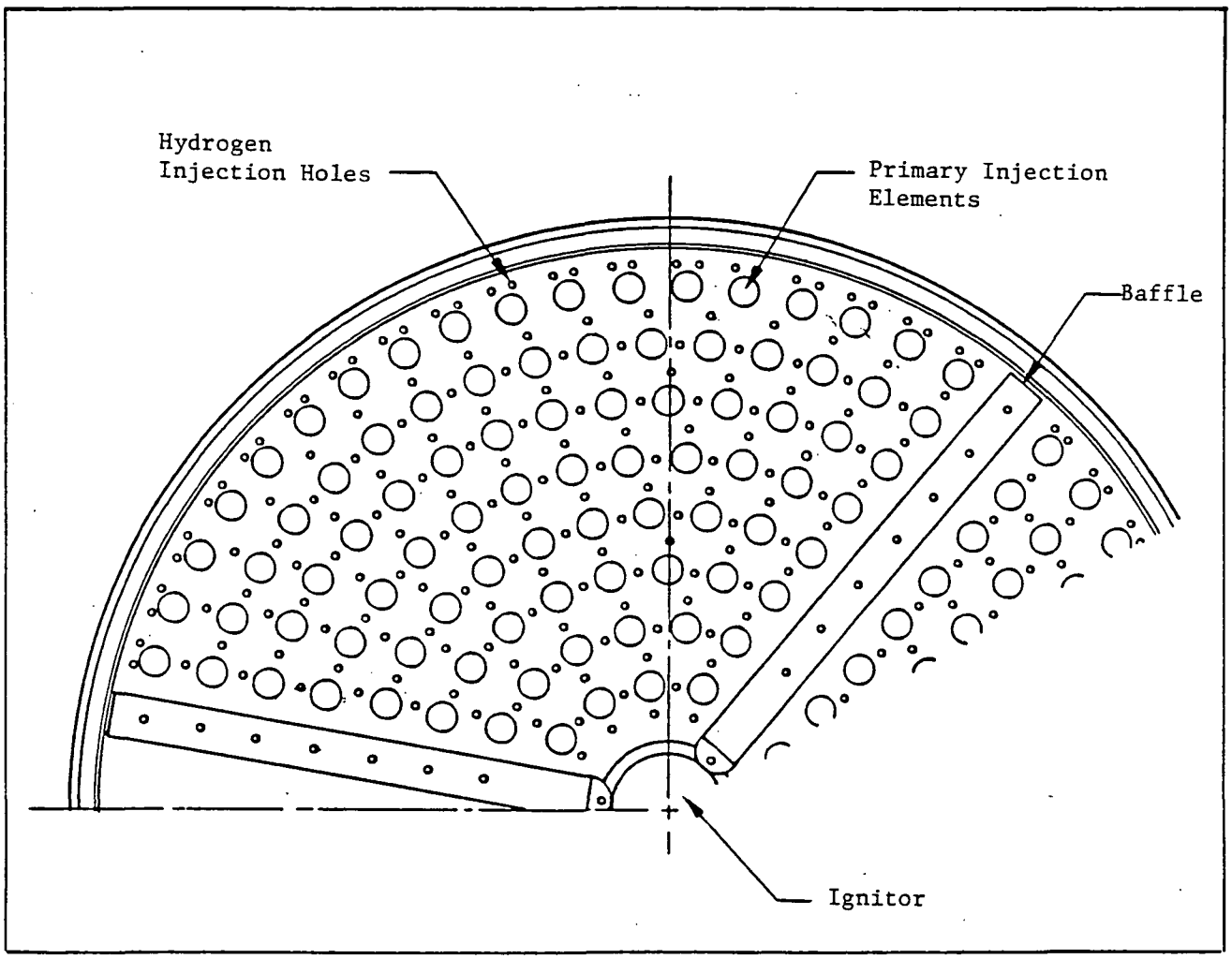


Fig. 2-4 Face Plate Showing One-Third of Preburner Distribution of Injection Holes and Elements

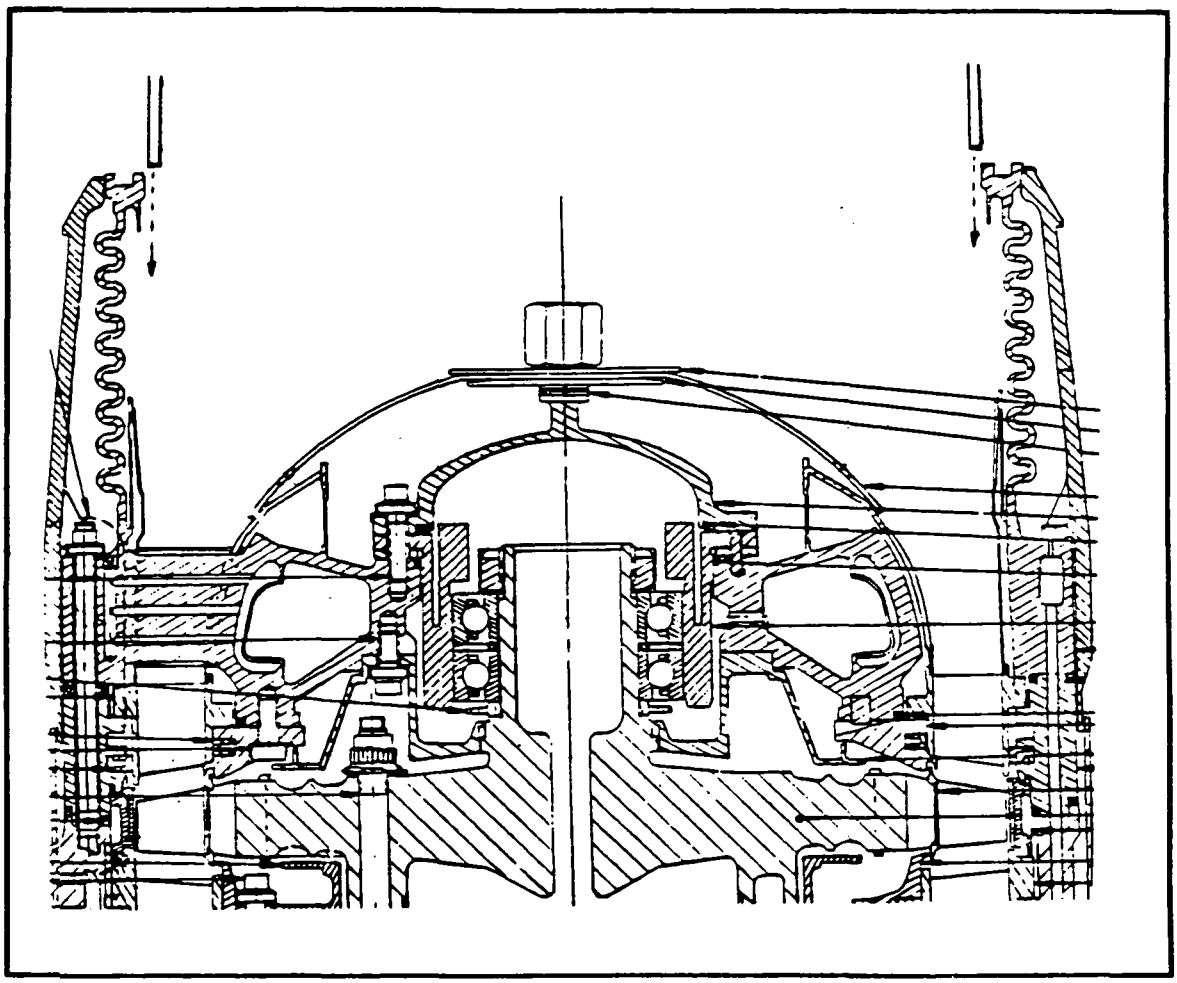


Fig. 2-5 Cross-Section of Lower Half Fuel Side Preburner Chamber Indicating at Top Left and Right Procedure for Insertion of Upper Preburner Liner Extension

Our approach is to use the symmetry created by the baffles and model only one-third of the geometry, and the large nut on top of the dome should be incorporated into the computational grid. Furthermore, in order that the most accurate analysis of the preburner be made for portions of the startup transient stage of its operation, a thorough study was made of the average chamber parameters for the first few seconds of operation. Data were obtained from the latest Digital Transient Model (DTM) results. The DTM is a one-dimensional lumped parameter model of the entire SSME which was originally developed by Rocketdyne and modified by D.C. Seymour of the Performance Analysis Branch of the Propulsion Lab at NASA-MSFC, to more closely match actual engine test data obtained at MSFC. These data were used to obtain, as accurate as possible, the face plate inlet conditions and downstream chamber exit conditions for use in the boundary treatment of a computational analysis.

3. RESULTS

3.1 HGM COMPUTATIONAL GRID CODE

3.1.1 HGM Geometry

A schematic representation showing the construction of the geometry which must be modeled is presented in Fig. 3-1. The 180-deg turnaround duct is displayed in a cutaway fashion to show the turn. Only half of the manifold is represented since, due to the plane of symmetry which divides the center transfer duct in half, only half need be computationally modeled.

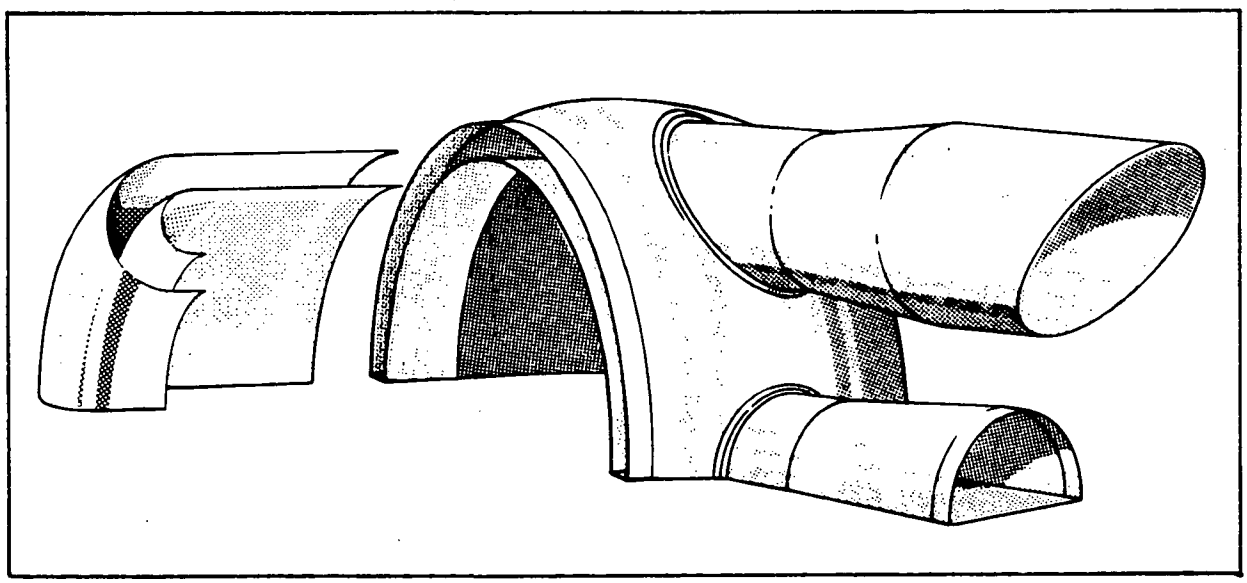


Fig. 3-1 Schematic Showing Components of HGM Geometry to be Modeled

The geometry was generated in five separate pieces or zones. The five zones are shown in Fig. 3-2. Each zone has a general hexahedral shape and in Fig. 3-2 most of the eight corner points of each are clearly marked. Figure 3-3 shows the Cartesian coordinate system relative to which the position of each node in the grid is referenced. Except for the hole perimeters in the outer wall of Zone 3 and the fairing at the entrances to Zones 4 and 5 the edges of each zone can be described as piecewise continuous segments composed of either straight lines or circular arcs. In addition, excluding the two special regions previously mentioned, the surfaces of each zone can be generated by rotating an edge about the appropriate axis. For Zones 1, 2, and 3 the X axis is the axis of revolution.

3.1.2 HGM Grid

Zone 1, the turnaround duct (TAD), is composed of 94 nodes in the streamwise direction, 72 nodes in the circumferential direction (0 to 180 deg), and 21 nodes across the duct between inner and outer surfaces. Two perspectives showing this nodal distribution are shown in Fig. 3-4.

Zones 2 and 3 make up the bowl section of the HGM and contain 59 nodes in the x-direction, from bowl entrance to rear of the bowl, 107 nodes in the circumferential direction and 21 nodes between the inner and outer surfaces. Distribution of nodes in the bowl is presented in Figs. 3-5 through 3-7.

Zones 4 and 5 comprise the right and half of the middle transfer duct portions of the HGM. The right duct has been generated with 29 nodes along the duct axis and 27 x 20 nodes in a cross section. The similar distribution in Zone 5 is 29 x 27 x 13. Surface and cross-section grids for these two zones are given in Figs. 3-8 and 3-9. A closeup of the surface mesh in the vicinity of the fairing for each transfer duct is contained in Fig. 3-10.

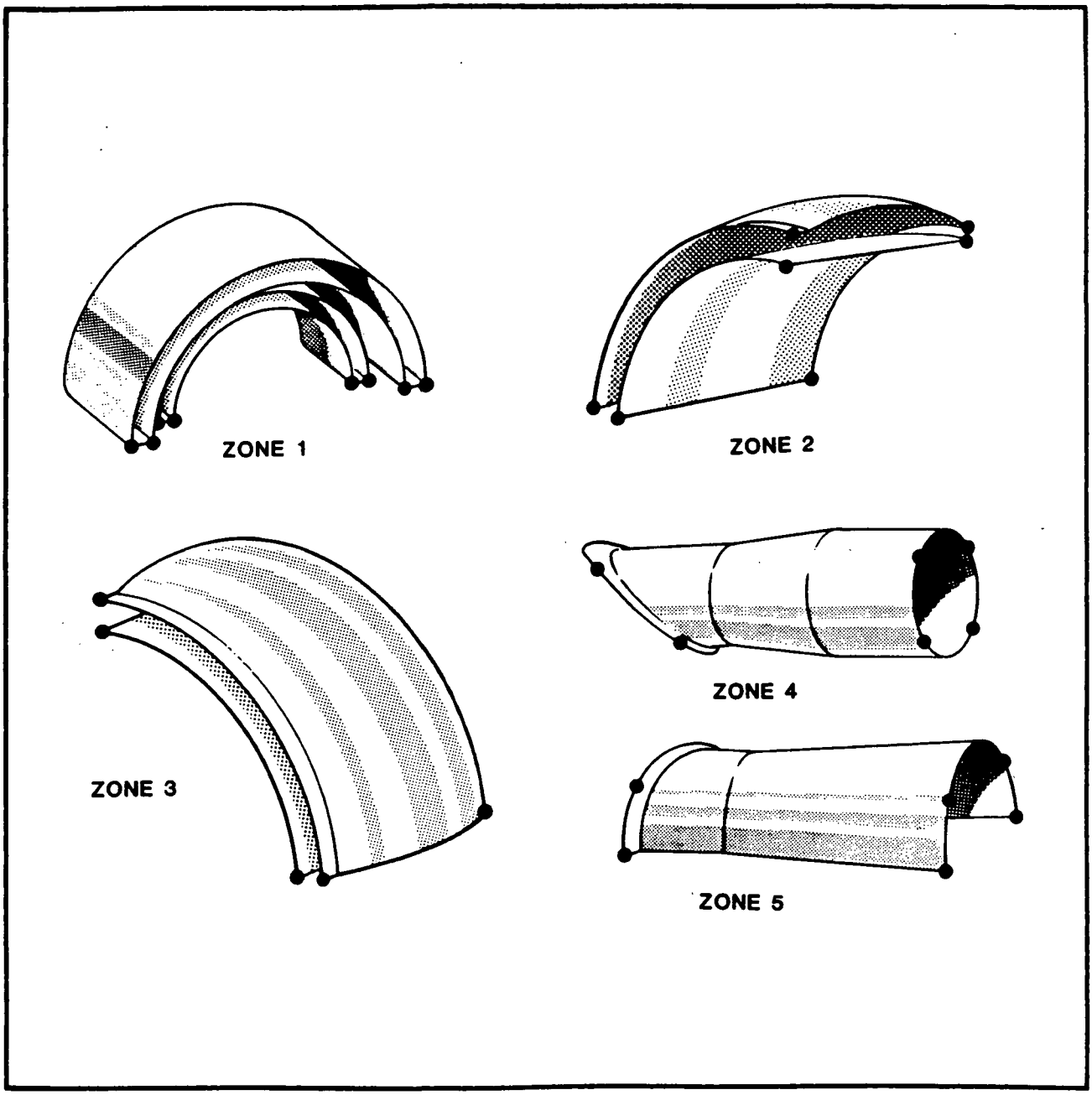


Fig. 3-2 Schematic Showing Five Zones into Which the Three-Duct HGM was Subdivided

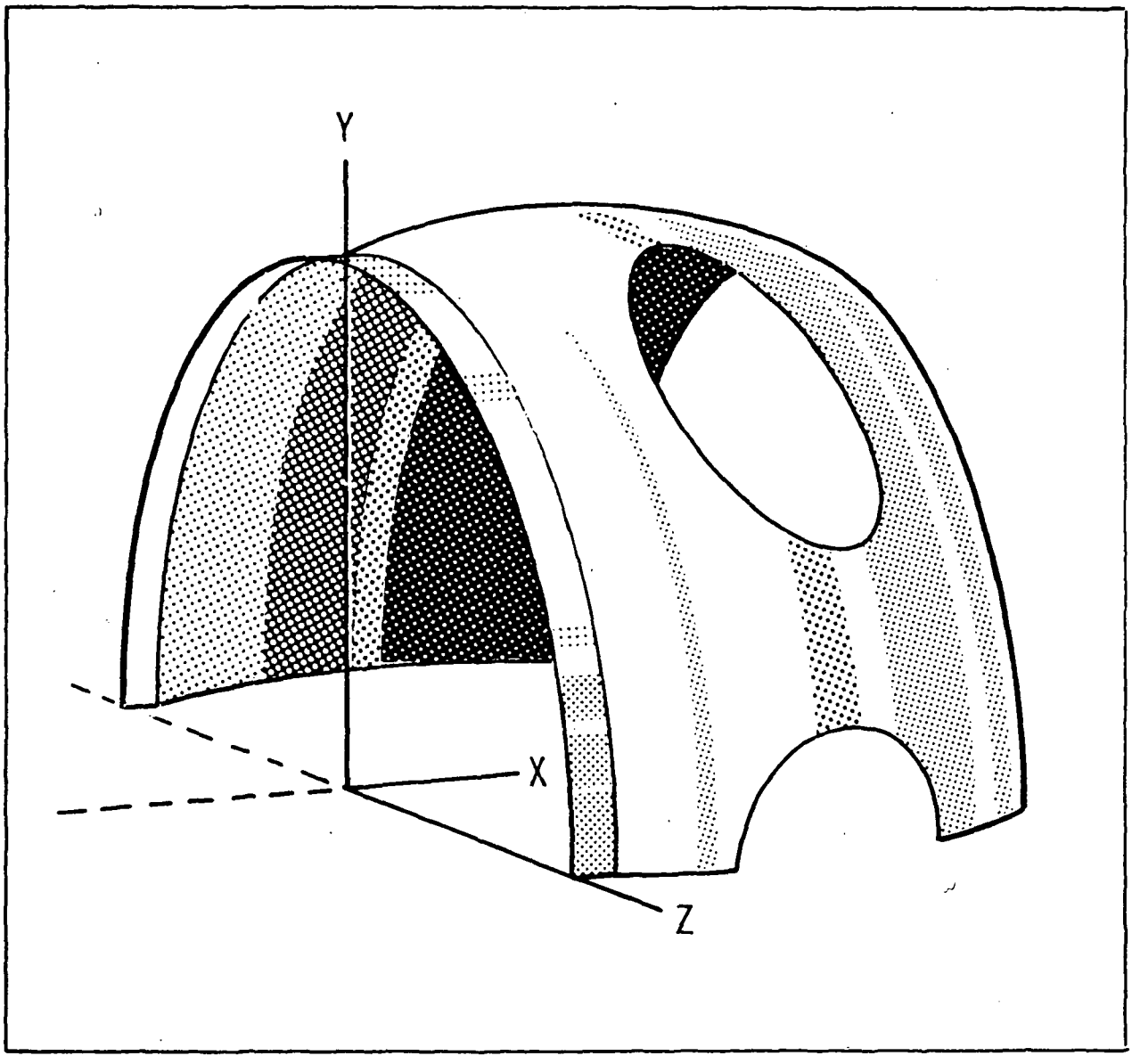


Fig. 3-3 Cartesian Coordinate System Relative to Which Position all Nodes are Referred

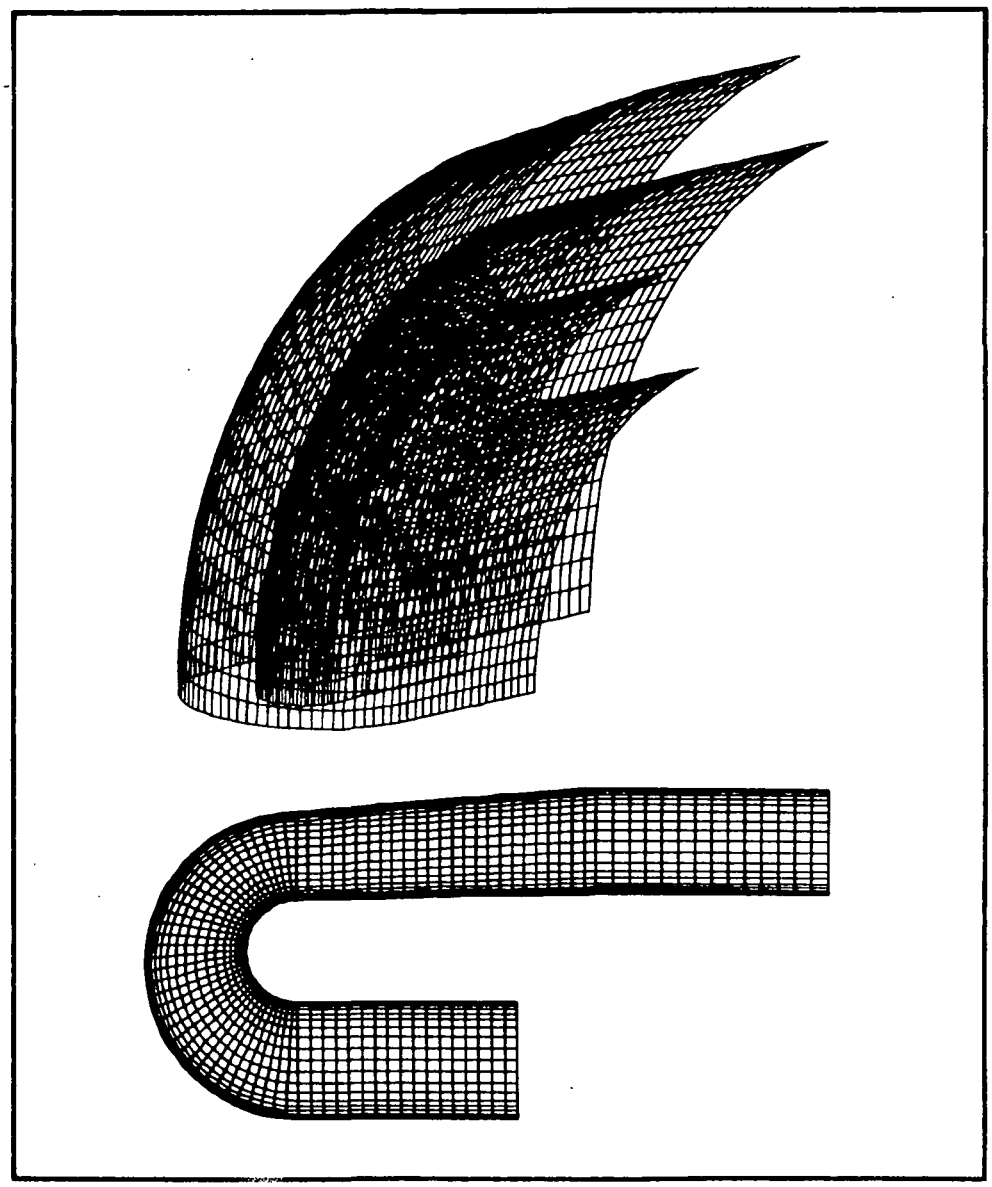


Fig. 3-4 Grid Plots of TAD Internal Grid (Bottom) and of Part of the Grid Distribution on Inner and Outer Surfaces

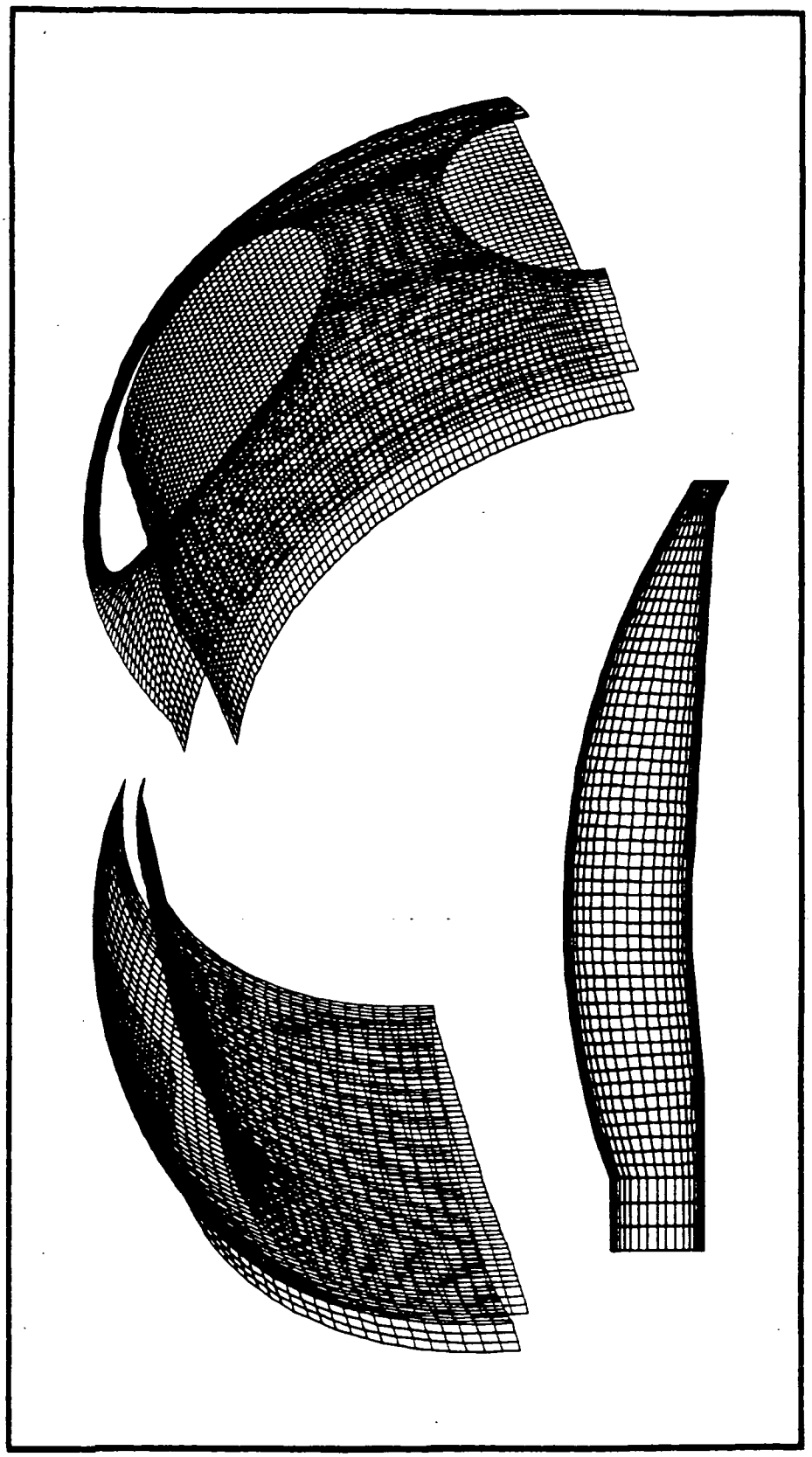


Fig. 3-5 Grid Plots for Inner and Outer Surfaces of Zones 2 and 3 as Well as Internal Grid at the Common Plane of Intersection

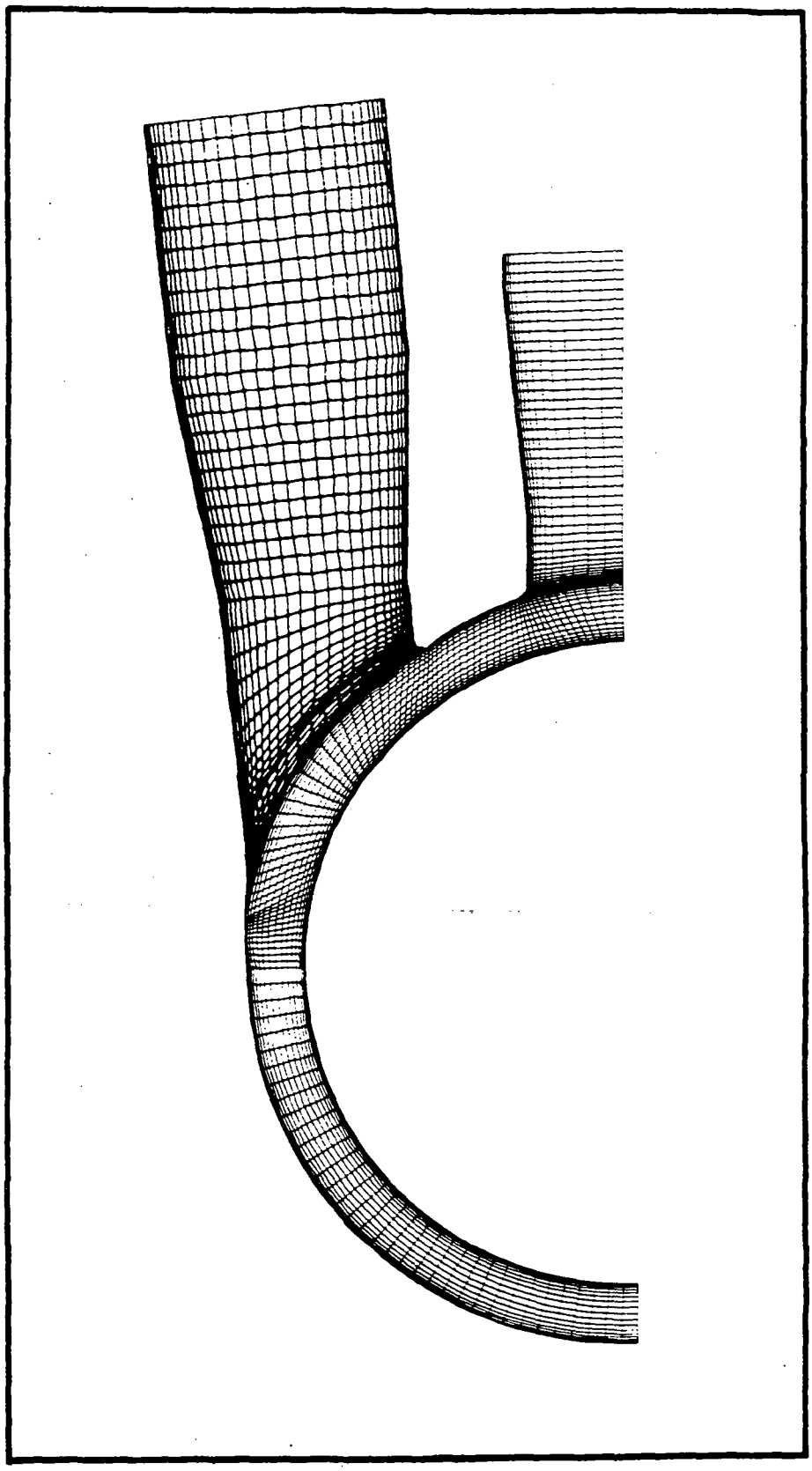


Fig. 3-6 Grid Plot Showing Node Distribution in Zones 2, 3, 4, and 5 on a Plane Which Passes Through the Center of Both Transfer Ducts

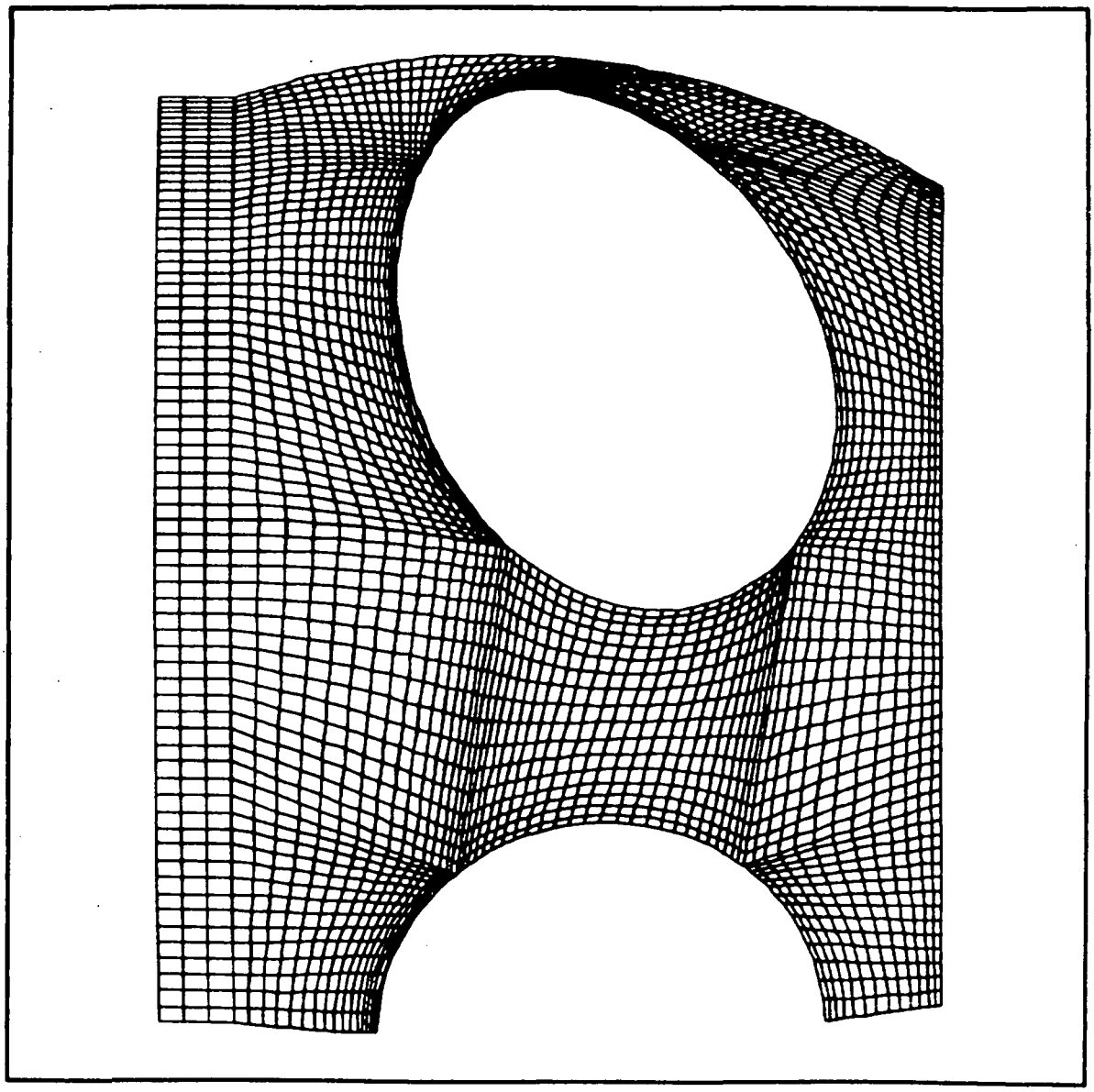


Fig. 3-7 Grid Plot of Computational Mesh on Outer Surface of Zone 3

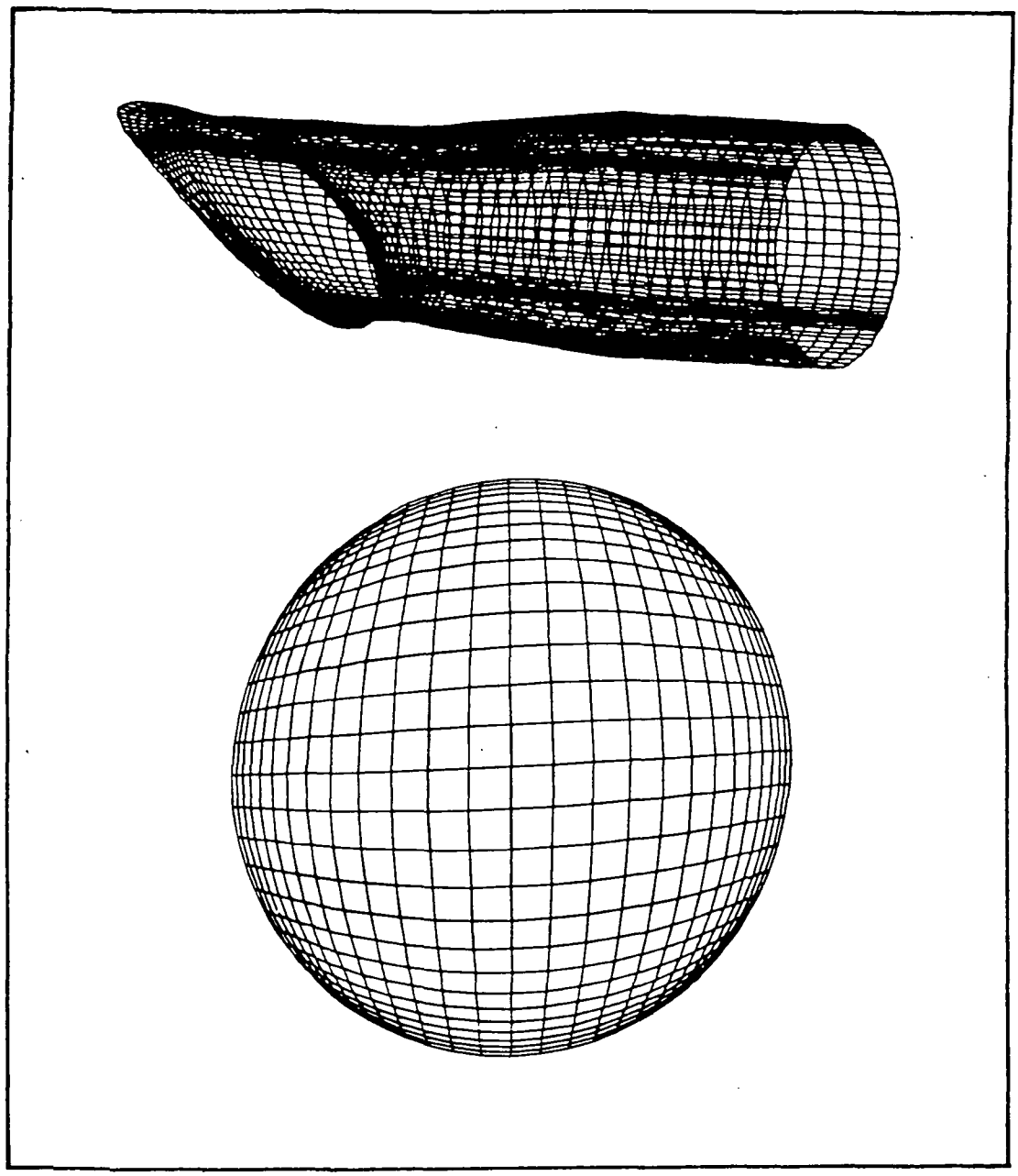


Fig. 3-8 Surface and Cross-Section Node Distribution for Top Transfer Duct

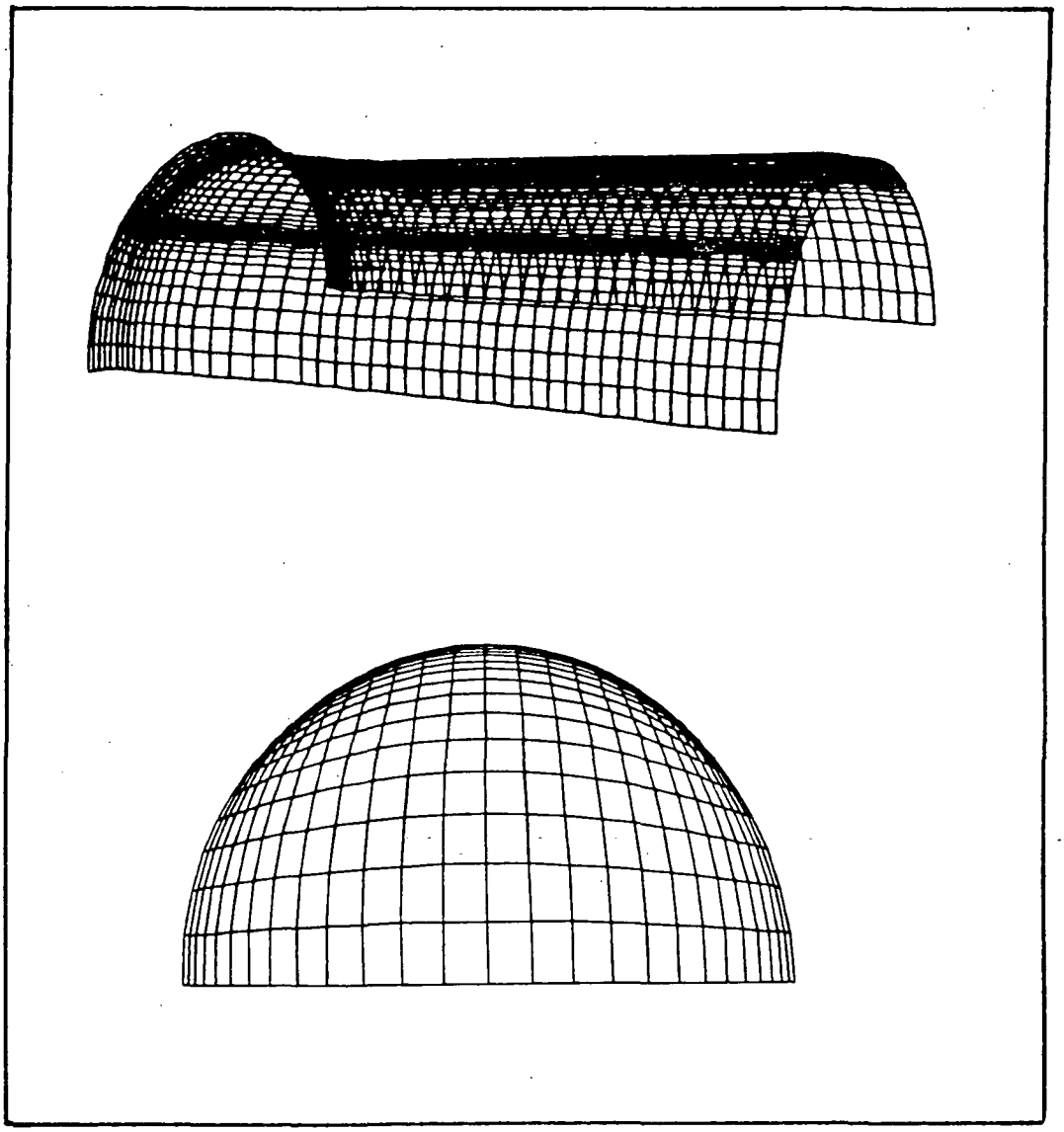


Fig. 3-9 Middle Transfer Duct Surface
and Cross-Section Nodal Distribution

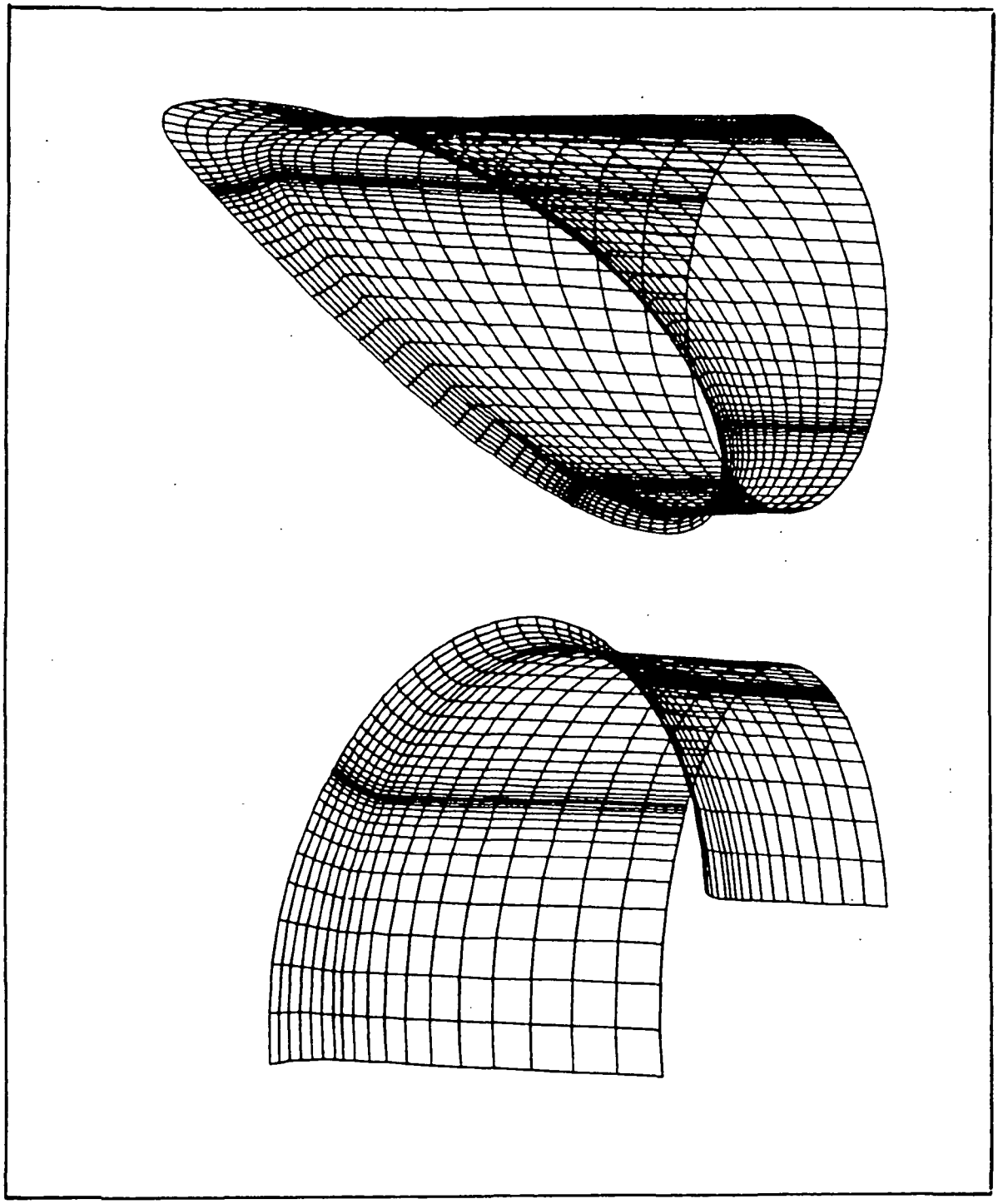


Fig. 3-10 Expanded View of Surface Mesh for Region of Fairing on Both Transfer Ducts

The internal grid resolution presented in the previous figures is adequate for a laminar viscous computation but would need to be modified for application to a turbulent computation. The procedure for doing this will be described in Section 3.1.4.

3.1.3 Computer Code

A source listing of the three-duct HGM geometry code is provided in Appendix C. A concise input guide and an input listing for the grids displayed in Figs. 3-4 through 3-9 is also included in Appendixes A and B, respectively.

In Fig. 3-11 a primary calling sequence flow chart is shown. A brief explanation of the function of each subroutine follows:

- INITIAL - Reads the first two lines of the input file, initializes coefficient arrays, and defines logical unit numbers and counters.
- INPUT - Reads the remainder of the input file and sets all parameters to be used in remaining subroutines.
- GRID - The controlling subroutine for the generation of the spatial coordinates of each node in each zone.
- ETABC - Calculates the values of η_1, η_2, η_3 along the I, J, and K directions for each hexahedral shaped section of each zone.
- EDGE - Determines the Cartesian coordinates for nodes along each edge of each side of each section of each zone using the bilinear/trilinear interpolation scheme.
- SURFACE - Determines the Cartesian coordinates of nodes on a three-dimensional surface using the trilinear interpolation scheme. Here, all outer and internal surface nodes are calculated from the previously determined edge distribution.
- OUTPUT - Provides printed output and stores geometry in File 20 for use in plotting or as input to an integration code.

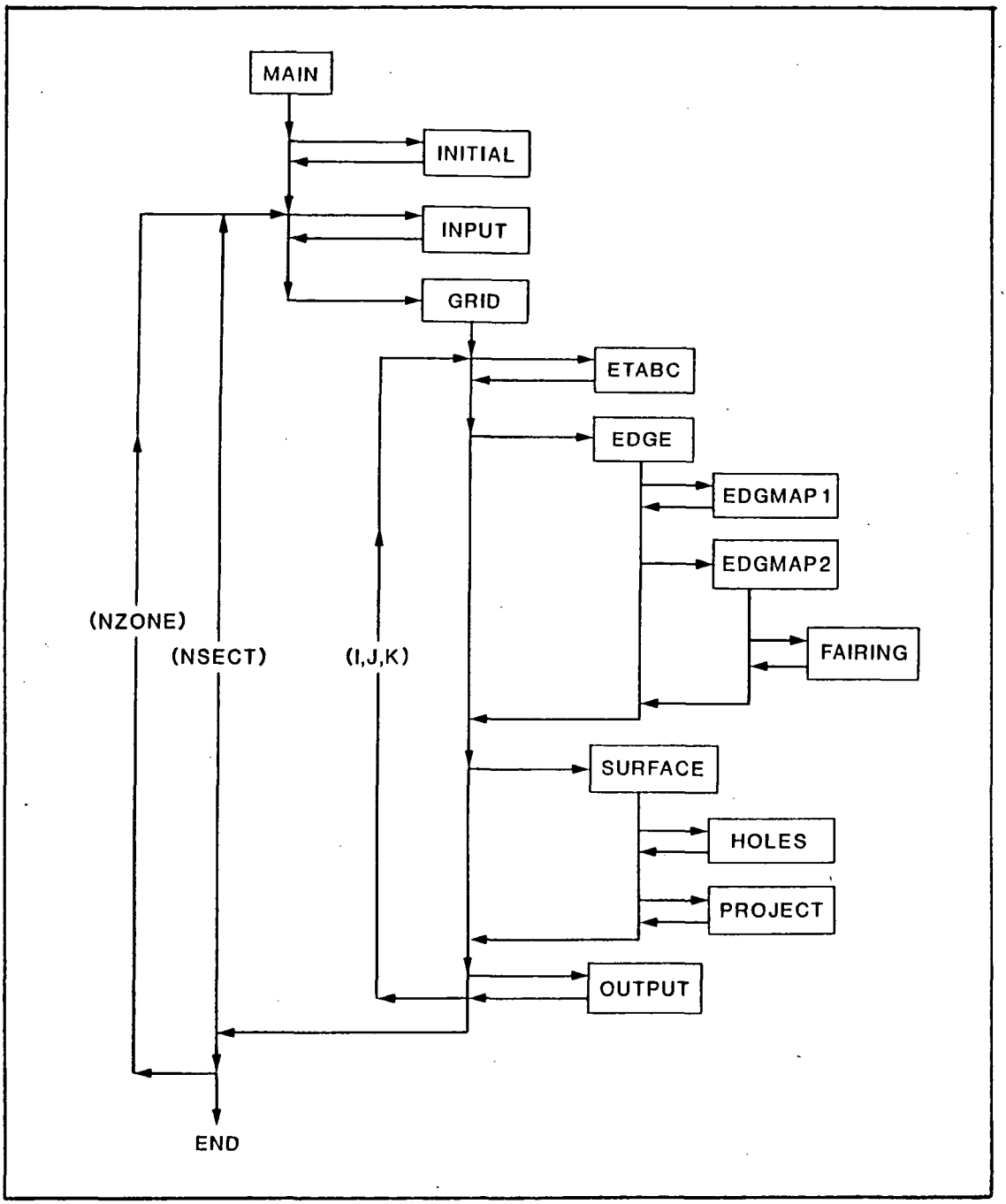


Fig. 3-11 Primary Calling Sequence

The output to File 20 is in the format to be input as a multi-grid geometry file to the PLOT3D plotting code.

The code is written in standard FORTRAN. However, the listing provided in Appendix C is a VAX 11/785 version and could contain some generic VAX statements which would have to be translated if used on other than a VAX.

3.1.4 Code Implementation

To implement the HGM geometry code as it appears in the listing of Appendix C, the input listing of Appendix B must correspond to logical Unit 5. Unit 6 must be assigned to the written output and Unit 20 to the geometry file which will contain the x, y, z coordinates for each node in each zone.

Each separate input file is labeled to indicate the zone being described by the following card images. These zone labels correspond to those shown in Fig. 3-2. Because of the number of nodes in each zone it is recommended that each zone be run separately. In fact as shown in Appendix B, each zone input is a separate file since all contain card types 1, 2, and 3. If more than one zone is to be run together (e.g., Zones 2 and 3), only one set of card types 1, 2, and 3 should appear at the top of the input file.

A detailed description of the input file to the code is provided in Appendix A. Modifications to the geometry can be facilitated by studying the input guide while observing both the input files and the detailed grid pictures presented in Figs. 3-3 through 3-10. Redistributions of nodes can be accomplished by making minor modifications to the input files. For example for a turbulent computation if the nodes in Zone 4 near the wall require redistribution closer to the wall then a change would need to be made to card type 9 on line 11 of the input file for that zone. The 7.0 appearing in the η_1 and η_3 positions could be changed to 10.0 (see page A-10). If more nodes were desired in a cross plane then the 27 and 29 on card type 8 could both be increased.

Dimensioning in the program has been kept to a minimum. The largest dimensioned arrays in the bulk of the code are NODENUM(10000) and NODE(5,10000). At the very end of all computations the PLOT3D file is generated. In this subroutine the x, y, z coordinate arrays are each dimensioned to 155,000. The 10,000 corresponds to twice the maximum number of nodes in a plane perpendicular to the marching direction (η direction input on card 7) for creating the geometry. The 155,000 must be equal to or greater than the number of nodes in the largest zone, which is the TAD. In Zones 1, 2, and 3 the direction of η_1 is from TAD entrance to bowl back wall; the η_2 direction is from inner to outer wall; and η_3 is directed from side opposite transfer ducts circumferentially. In zones 4 and 5, η_3 increases in the streamwise direction from bowl outward, and $\eta_1 \times \eta_2$ form the cross planes in each duct.

Note that the code is designed to output each zone of the HGM so that each has one cross plane in common with the preceding zone(s). This must be remembered for incorporating the grid into a flowfield solver code. The geometry must be integrated in a multi-block or multi-zone fashion. If the computer available has large enough core memory or if it is a large virtual machine then Zones 2 and 3 can easily be combined into one larger zone since at all common planes the nodal positions match exactly. However, the TAD zone matches exactly to Zone 2 but only every other circumferential plane in Zone 3 matches with a corresponding plane in the TAD. This mismatch can be corrected by making appropriate minor modifications to the input file for Zone 1.

3.2 PREBURNER ANALYSIS

An accurate analysis of the fluid dynamic environment in the preburner during the startup transient portion of its operation demands careful treatment of the inflow and outflow boundary conditions as well as the most up-to-date chemistry model for the reacting components in the flow. Results of a comprehensive study of these two items is presented in the following two sections.

3.2.1 Inflow and Outflow Boundary Treatment

Data were obtained in the form of printed output from the most recent DTM analysis of the SSME startup transient conditions. This included injector fuel and oxidizer flow rates, ignition fuel, and oxidizer flow rates. From this, average chamber temperature and static pressure data tables were generated. These data are displayed graphically in Figs. 3-12 through 3-16. At any instant in time during the 5-sec startup process, these data provide information necessary to specify mass and composition flow rates at the inlet plane as well as downstream static pressure at the exit plane for performing a flowfield computation.

During any selected time interval, downstream static pressure would be obtained from a table look-up of the data shown in Fig. 3-16. Total flow rates are obtainable from the remaining data, but the specific strategy for distributing the mass flow over the face plate inlet plane requires careful analyses. Observation of Fig. 2-4 reveals that there is clearly a nonuniform distribution of mass injection. A practical approach which allows for a reasonable approximation to the physical situation would be to divide the inlet computational plane into 18 regions; 17 concentric annular regions forming the injection elements and holes and one ignition region. The 17 injection regions correspond to the 17 rings of holes and elements and holes.

Nine of these annular regions contain only hydrogen injection while in the remaining eight both hydrogen and oxygen injection occurs, and within each region fluid is injected nonuniformly. To most accurately model each region fluid should be injected with the proper momentum from computational nodes which form an area which is approximately the same as the effective area for that region. The remaining nodes in that region must have a zero velocity fixed boundary condition. The effective area is less than the actual combined areas of holes and elements.

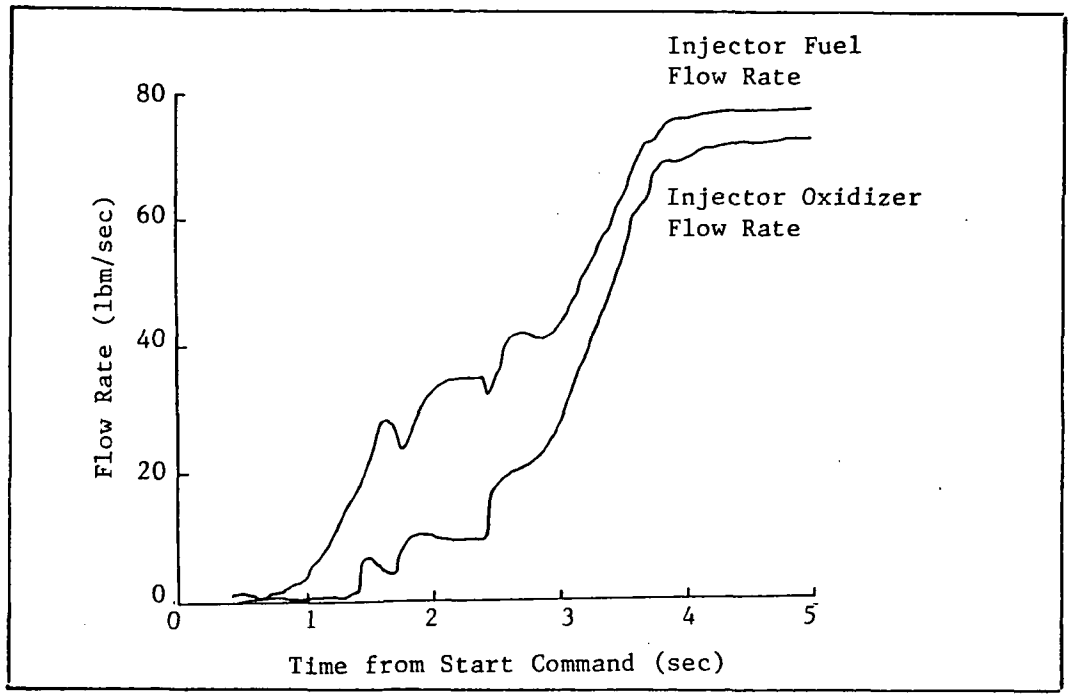


Fig. 3-12 Fuel Preburner Flow Rates Taken from Digital Transient Model (DTM)

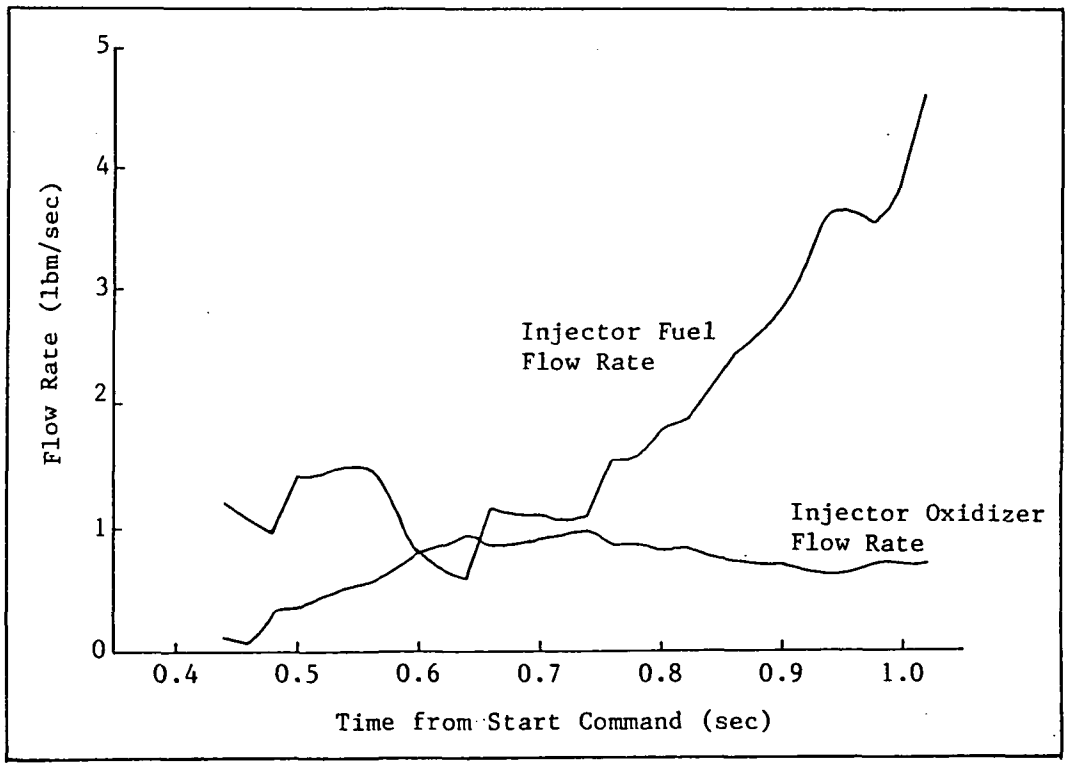


Fig. 3-13 Fuel Preburner Flow Rates from DTM During Early Transient

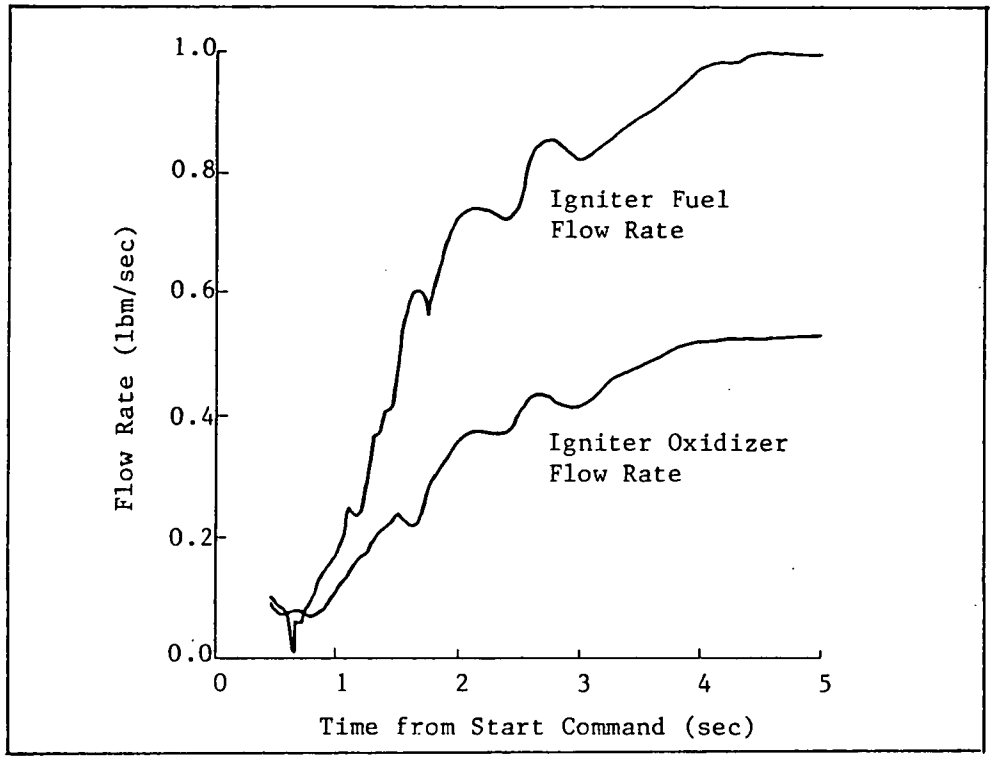


Fig. 3-14 Fuel Preburner Igniter Flow Rates from Digital Transient Model (DTM)

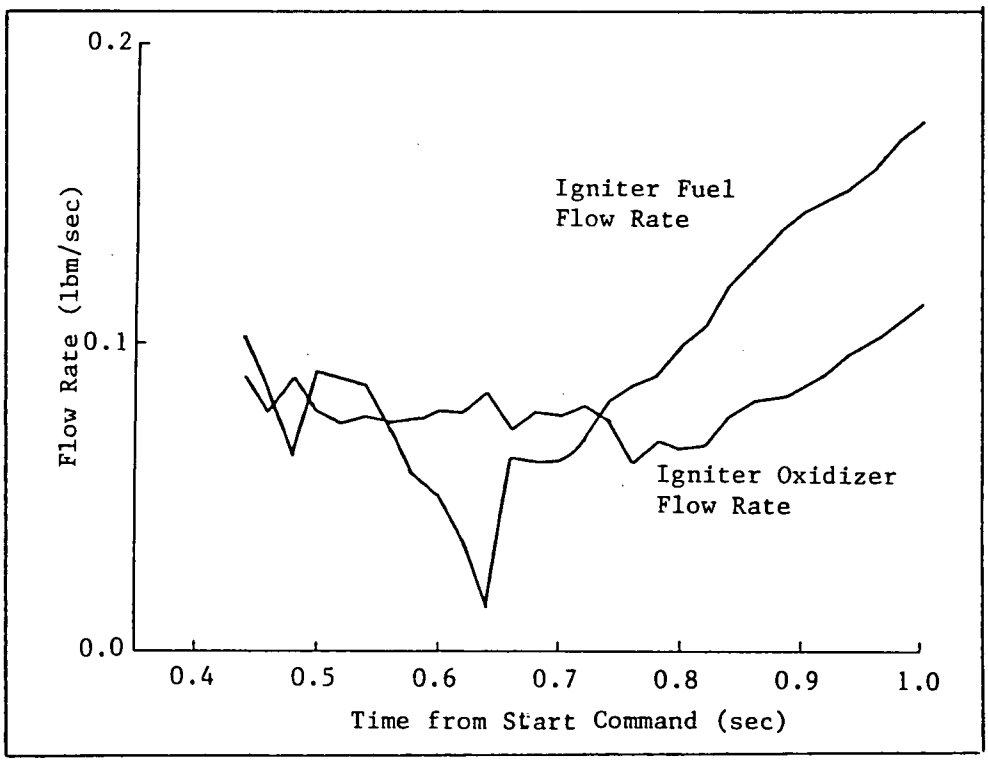


Fig. 3-15 Fuel Preburner Igniter Flow Rates from DTM During Early Transient

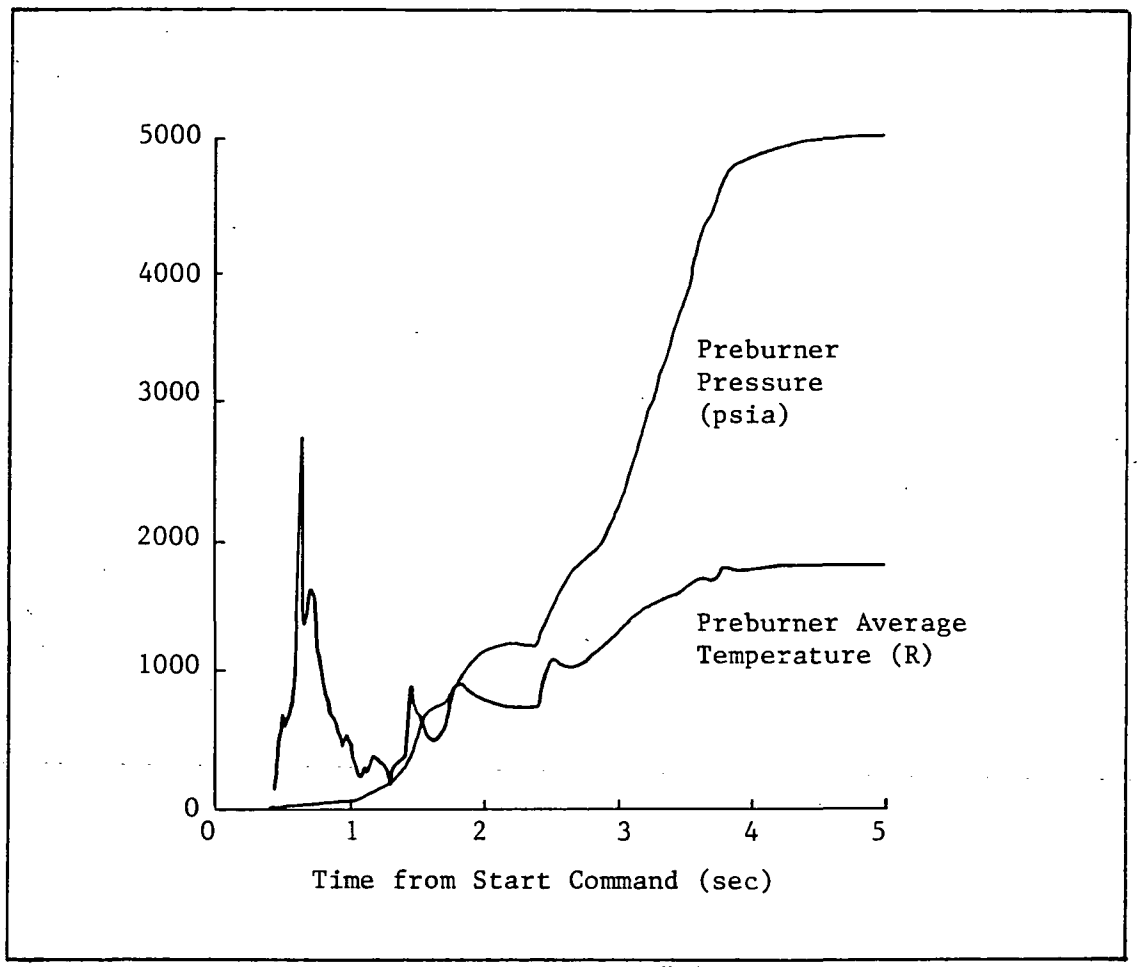


Fig. 3-16 Fuel Preburner Average Chamber Pressure and Temperature During Start-Up Transient (Taken from DTM)

The mass flow rate, $\Delta M/\Delta t$, through an area A of fluid having mass density of ρ and inlet speed V can be expressed as

$$\frac{\Delta M}{\Delta t} = \rho VA$$

The ratio of dynamic pressure to pressure loss through the opening defines a discharge coefficient C_d given by

$$C_d = \frac{1}{2} \rho V^2/\Delta p$$

Combining these two equations one expression for the product of A and C_d is obtained, which corresponds to the effective area.

$$AC_d = \frac{\Delta M/\Delta t}{0.67 (\rho \Delta p)^{1/2}}$$

In this equation the 0.67 includes unit conversion factors so that AC_d is determined in square inches. Data obtained from the SSME Power Balance Program, Revision F, provide a Δp for the entire face plate at FPL. Combining this with the appropriate mass flow rate at FPL one obtains an AC_d for the entire face plate of 3.32 in^2. This should be equal to the sum of the AC_d products for all of the fuel holes and the annular injection elements. The actual combined hole area is 0.792 in^2 and that of the combined annular fuel elements is 3.59 in^2. Assuming a C_d of 1.0 for the holes we thus determine C_d for the fuel elements to be 0.704. Because of the geometry of the oxygen injection path through the center of the injection elements it is reasonable to expect that the C_d for these holes is essentially unity.

With this information an effective area which should be modeled by the computational grid for each annular inlet plane region can be compiled and is presented in Table 3-1. Note that if a mass flow rate is computed for each region and is input uniformly over that part of the grid the stream will have the incorrect momentum. It is absolutely necessary that the areas of the

Table 3-1 EFFECTIVE AREAS (IN SQUARE INCHES) OF EACH ANNULAR ZONE (360 DEGREES) OF PREBURNER INJECTOR FACE-PLATE FOR COMPUTATIONAL MODELING OF INLET PLANE

Zone	Fuel Elements	Holes	Baffles	Oxidizer Injection Area
1		0.075745		
2	0.54639		0.0093	0.335934
3		0.044639		
4	0.48568	0.041849	0.0093	0.298608
5		0.039060		
6	0.42497	0.036270	0.0093	0.261282
7		0.033479		
8	0.36426	0.030690	0.0093	0.223956
9		0.041780		
10	0.30355	0.037599	0.0093	0.186630
11		0.033420		
12	0.24284	0.029243	0.0093	0.149304
13		0.030750		
14	0.18213	0.025620	0.00853	0.111978
15		0.020500		
16	0.12142	0.015373	0.00615	0.074652
17		0.020500	0.02214	

regions of mass injection in each ring on the inlet plane closely match the appropriate effective area. Thus many of the inlet plane nodes in the grid will be "turned off."

3.2.2 Chemistry Model

During the startup transient it is not clear how much and where combustion or burning occurs. It cannot be assumed that during this time interval the chemistry is completed within a short distance of the face plate. Much research was performed to ascertain the most appropriate chemistry model to employ for computing a realistic flow-kinetic picture for this problem.

It was determined that a non-equilibrium chemistry model be used to model the details of the kinetics. The eight reactive species are H_2, O_2, H_2O, H_2O_2, H, O, OH, and HO_2. A 20-reaction set was selected, eight of which apply to ignited hot portions of the flow field and 12 of which model ignition and cool flows. Tables 3-2 and 3-3 list these reactions along with their rate coefficients. Literature references are also included for completeness.

A reduced set of reactions can be substituted for these 20-reactions for a less complete treatment. How much of the physics is lost, if any, can only be discovered by computing a test problem using both sets. This has not been done.

The reduced set is obtained as follows:

1. Use Reactions 1 through 6 in Table 3-2
2. Use Reaction 9 in Table 3-3
3. Rewrite Reaction 10 as

$$H + O_2 + M \rightarrow 0.5\ H_2 + O_2 + M$$

The eighth reaction is forward only, and the rate coefficient should be set equal to two thirds that for Reaction 10. Thus the minimum recommended chemistry treatment should contain eight reactions and six species.

Table 3-2 KINETIC REACTIONS (1 THROUGH 8)

N Body Reaction (Reversible)	Rate Coefficient $(cm^3 particle^{-1})^{(N-1)} sec^{-1}$ units	Ref.
1. $OH + H_2 \rightarrow H_2O + H$	$1.7 \times 10^{-16} T^{1.6} \exp(-3,300/RT)$	1, 2
2. $H + O_2 \rightarrow OH + O$	$2.5 \times 10^{-7} T^{-0.9} \exp(-17,330/RT)$	3, 4
3. $O + H_2 \rightarrow OH + H$	$1.8 \times 10^{-20} T^{2.8} \exp(-5,920/RT)$	4
4. $OH + OH \rightarrow H_2O + O$	$1.4 \times 10^{-20} T^{2.7} \exp(1,800/RT)$	5, 6
5. $H + OH + M \rightarrow H_2O + M$	$2.4 \times 10^{-26} T^{-2}$	7, 8
6. $H + H + M \rightarrow H_2 + M$	$2.0 \times 10^{-30} T^{-1}$ $(M \neq H_2)$	9, 2
	$2.8 \times 10^{-31} T^{-0.6}$ $(M = H_2)$	2
7. $H + O + M \rightarrow OH + M$	$4.0 \times 10^{-29} T^{-1}$	10
8. $O + O + M \rightarrow O_2 + M$	$6.0 \times 10^{-35} \exp(1,800/RT)$	11

Table 3-3 KINETIC REACTIONS (9 THROUGH 20)

N Body Reaction (Reversible)	Rate Coefficient $(cm^3 particle^{-1})^{(N-1)} sec^{-1}$ units	Ref.
9. $H_2 + O_2 \rightarrow OH + OH$	$2.8 \times 10^{-11} \exp(-48,100/RT)$	12
10. $H + O_2 + M \rightarrow HO_2 + M$	$2.0 \times 10^{-30} T^{-0.8}$	13
11. $HO_2 + OH \rightarrow H_2O + O_2$	$1.6 \times 10^{-11} \exp(1,000/RT)$	14
12. $HO_2 + H \rightarrow H_2 + O_2$	$4.2 \times 10^{-11} \exp(-700/RT)$	9, 2
13. $HO_2 + H \rightarrow OH + OH$	$1.9 \times 10^{-10} \exp(-1,000/RT)$	16
14. $HO_2 + O \rightarrow OH + O_2$	$3.3 \times 10^{-11} \exp(400/RT)$	17
15. $HO_2 + HO_2 \rightarrow H_2O_2 + O_2$	$3.2 \times 10^{-12} \exp(2,300/RT)$	15
16. $HO_2 + H_2 \rightarrow H_2O_2 + H$	$3.2 \times 10^{-12} \exp(-21,000/RT)$	15
17. $H_2O_2 + OH \rightarrow HO_2 + H_2O$	$3.7 \times 10^{-12} \exp(-500/RT)$	18
18. $H_2O_2 + H \rightarrow OH + H_2O$	$1.7 \times 10^{-11} \exp(-3,600/RT)$	2
19. $H_2O_2 + O \rightarrow OH + HO_2$	$8.7 \times 10^{-13} \exp(-3,600/RT)$	20
20. $H_2O_2 + M \rightarrow OH + OH + M$	$1.2 \times 10^{-7} \exp(-46,300/RT)$	19

4. SUMMARY AND CONCLUSIONS

A three-duct SSME Hot Gas Manifold geometry code has been developed for use by the computational mechanics staff of NASA-MSFC. This report describes the methodology of the program, makes recommendations on its implementation, and provides an input guide, and input deck listing, and a source code listing. The code listing is strewn with an abundance of comments to assist the user in following its development and logic. A working source deck will be provided to NASA-MSFC on the EADS network upon request.

A thorough analysis has been made of the proper boundary conditions and chemistry kinetics necessary for an accurate computational analysis of the flow environment in the SSME fuel side preburner chamber during the initial startup transient. Pertinent results have been presented to facilitate incorporation of these findings into an appropriate CFD code. The computation must be a turbulent computation, since the flow field turbulent mixing will have a profound effect on the chemistry. Because of the additional equations demanded by the chemistry model it is recommended that for expediency a simple algebraic mixing length model be adopted.

Performing this computation for all or selected time intervals of the startup time will require an abundance of computer CPU time regardless of the specific CFD code selected.

5. REFERENCES

1. Zellner, R., J. Phys. Chem., Vol. 83, p. 18 (1979).

2. Warnatz, J., "Survey of Rate Coefficients in the C/H/O System," in W.C. Gardiner (ed.), Chemistry of Combustion Reactions, Springer, New York, 1984, pp. 197-360.

3. Miller, J.A., "Nonstatical Effects and Detailed Balance in Quasiclassical Trajectory Calculations of the Thermal Rate Coefficient for $O + OH \rightarrow O_2 + H$," J. Chem. Phys., Vol. 84, 1986, pp. 6170-6177.

4. Cohen, N., and K.R. Westberg, "Chemical Kinetic Data Sheets for High-Temperature Chemical Reactions," J. Phys. Chem., Reference Data, Vol. 12, 1983, pp. 531-562.

5. Fontijn, A., and R. Zellner, "Influence of Temperature on Rate Coefficients of Bimolecular Reactions," in A. Fontijn and M.A.A. Clyne (eds.), Reactions of Small Transient Species - Kinetics and Energetics, Academic Press, New York, 1983, pp. 1-61.

6. Wagner, G., and R. Zellner, "Temperature Dependence of the Reaction $OH + OH \rightarrow H_2O + O$," Ber. Bunsenges, Phys. Chem., Vol. 85, 1981, pp. 1122-1128. (As reviewed in Ref. 5, these authors' results for Reaction 4 were written as $k_4 = 9.06 \times 10^{-13} \exp(\quad)$ over the range 250-2000 K. This expression has been refit here to the format $k = AT^n \exp -(E/RT)$ for use in computations. The other reviews examined to not incorporate data beyond 1977.)

7. Luther, K., and J. Troe, "Influence of Temperatures on Unimolecular and Termolecular Reactions," in A. Fontijn and M.A.A. Clyne (eds.), Reactions of Small Transient Species - Kinetics and Energetics, Academic Press, New York, 1983, pp. 63-104.

8. Troe, J., "Atom and Radical Recombination Reactions," in Ann. Rev. Phys. Chem., Vol. 29, Annual Reviews, Palo Alto, 1978, pp. 223-250. (Low pressure recombination results in Ar from Ref. 8 - reviewed further in Ref. 8 have been utilized here to obtain the recommended expression.)

9. Baulch, D.L., Drysdale D.G. Horne, and A.C. Loyd, Evaluated Kinetic Data for High Temperature Reactions. Volume 1 - Homogeneous Gas Phase Reactions of the H_2 - O System, Butterworths, London, 1972.

5. REFERENCES (Continued)

10. Schofield, K., "Evaluated Chemical Kinetic Rate Constants for Various Gas Phase Reactions," J. Phys. Chem., Reference Data, Vol. 2, 1973, pp. 25-84. (Schofield's recommendation that $k = 2 \times 10^{-32}$ over the 1000-3000 K range has been refit here to a T^{-1} dependence anchored at 2000 K. There is no more recent input for this poorly known rate coefficient.)

11. Baulch, D.L., D.D. Drysdale, J. Duxbury, and S.J. Grant, Evaluated Kinetic Data for High Temperature Reactions, Vol. 3: Homogeneous Gas Phase Reactions of the $O_2 - O_3 - H_2$ System and of Sulphur Containing Species, Butterworths, London, 1976.

12. Jachimowski, C.J., and W.M. Houghton, "Shock Tube Study of the Initiation Process in the Hydrogen - Oxygen Reaction," Combustion and Flame, Vol. 17, 1971, pp. 25-30. (There are no more recent determinations of this rate coefficient.)

13. Baulch, D.L., R.A. Cox, R.F. Hampson, J.A. Kerr, J. Troe, and R.T. Watson, "Evaluated Kinetic and Photochemical Data for Atmospheric Chemistry," J. Phys. Chem., Reference Data, Vol. 9, 1980, pp. 295-468.

14. Data reviewed in Ref. 15 by Kaufman and Sherwell indicate a preferred room temperature value f 6.6×10^{-11} at low pressures and 10.7×10^{-11} at pressures nearer latm; also, flame studies and shock tube work give lower results (2.0×10^{-11} at 2130 K) at higher temperatures. These data are reconciled by the interim expression recommended here.

15. Kaufman, M., and J. Sherwell, "Kinetics of Gaseous Hydroperoxyl Radical Reactions," in Prog. Reaction Kinetics, Vol 12, Pergamon, London, 1983, pp. 1-53.

16. The expression recommended for $k_{12} + k_{13}$ at room temperature, i.e., 4.8×10^{-11}, utilizing the indicated volume for k_{12}.

17. DeMore, W.B., J.J. Margitan, M.J. Molina, R.T. Watson, D.M. GOlden, R.F. Hampson, M.J. Kurylo, C.J. Howard, and A.R. Ravishankara, Evaluation Panel, "Chemical Kinetics and Photochemical Data for Use in Stratospheric Modeling," NASA CR-176198, JPL Publication 85-37, July 1985.

18. The expression given here for k_{17} is based on a simple Arrhenius fit to the value $k_{17} = 1.6 \times 10^{-12}$ at room temperature recommended by Kaufman and Sherwell (Ref. 15 and the measured ratio of 3.2 ± 0.6 for k_{17}/k_1 at 773 K by Baldwin et al. (Ref. 19), yielding $k_{17} = 2.7 \times 10^{-12}$ at 773 K, utilizing the recommended value for k_1.

5. REFERENCES (Concluded)

19. Baldwin, R.R., C.E. Dean, M.R. Honeyman, and R.W. Walker, "The Mutual Reaction of HO_2 Radicals and the Rate Constants of the $H_2 + O_2$ Reaction," J. Chem. Soc., Faraday Trans. 1, Vol. 80, 1984, pp. 3187-3194.

20. The expression given here for k_{19} is based on a simple Arrhenius fit to the mean of the values for k_{19} (2.3×10^{-19} and 1.7×10^{-19}) at room temperature recommended by Baulch et al. (Ref. 21) and by DeMore et al. (Ref. 17), and the measured ratio of 10.4 for k_{19}/k_2 at 773 K by Baldwin et al. (Ref. 19) yielding $k_{19} = 8.3 \times 10^{-14}$ at 773 K, utilizing the recommended value for k_1.

21. Baulch, D.L., R.A. Cox, P.J. Crutzen, R.F. Hampson, J.A. Kerr, J. Troe, and R.T. Watson, "Evaluated Kinetic and Photochemical Data for Atmospheric Chemistry: Supplement 1," J. Phys. Chem., Reference Data, Vol. 11, 1982, pp. 327-496.

Appendix A
GEOMETRY INPUT GUIDE

Appendix A

INTRODUCTION

The geometry input guide is presented in two sections: (1) a definition of terminology commonly used for inputting and describing the geometry, and (2) a summary of card types used to input the geometry and a detailed description of the associated parameters and their input values.

We begin with an overview of how to apply the program. The flowfield domain is divided into zones in order to simplify the input necessary to describe the complicated geometry. Each zone contains its own coordinate system. A zone consists of at least one section. A multiple section zone is used to describe Zones 4 and 5. Sections may be added in only one internal coordinate direction which is called the marching direction since this is the direction in which the grid is created. Each section is described using points, edges, and surfaces. An edge may consist of from one to ten segments. A segment or a surface may require additional input depending on its type.

The second section presents a detailed description of the input parameters. Each card type is listed in the order of input with its associated parameters. Each parameter is identified as to its usage in the program with the options of each shown. Reference to Fig. A-1 or Table A-1 may be necessary to explain some of the input parameters and their order of input. All of the card types are not necessarily input for a specific zone.

Card type 9 may be used when other than an equal distribution of nodes is desired, etc. Whereas card type 13 is necessary if there will be more than one segment per edge. And cards type 10 and 11 are used if additional information is needed to describe a segment or a surface. Certain of the input parameters on early cards dictate which of the later cards are read in.

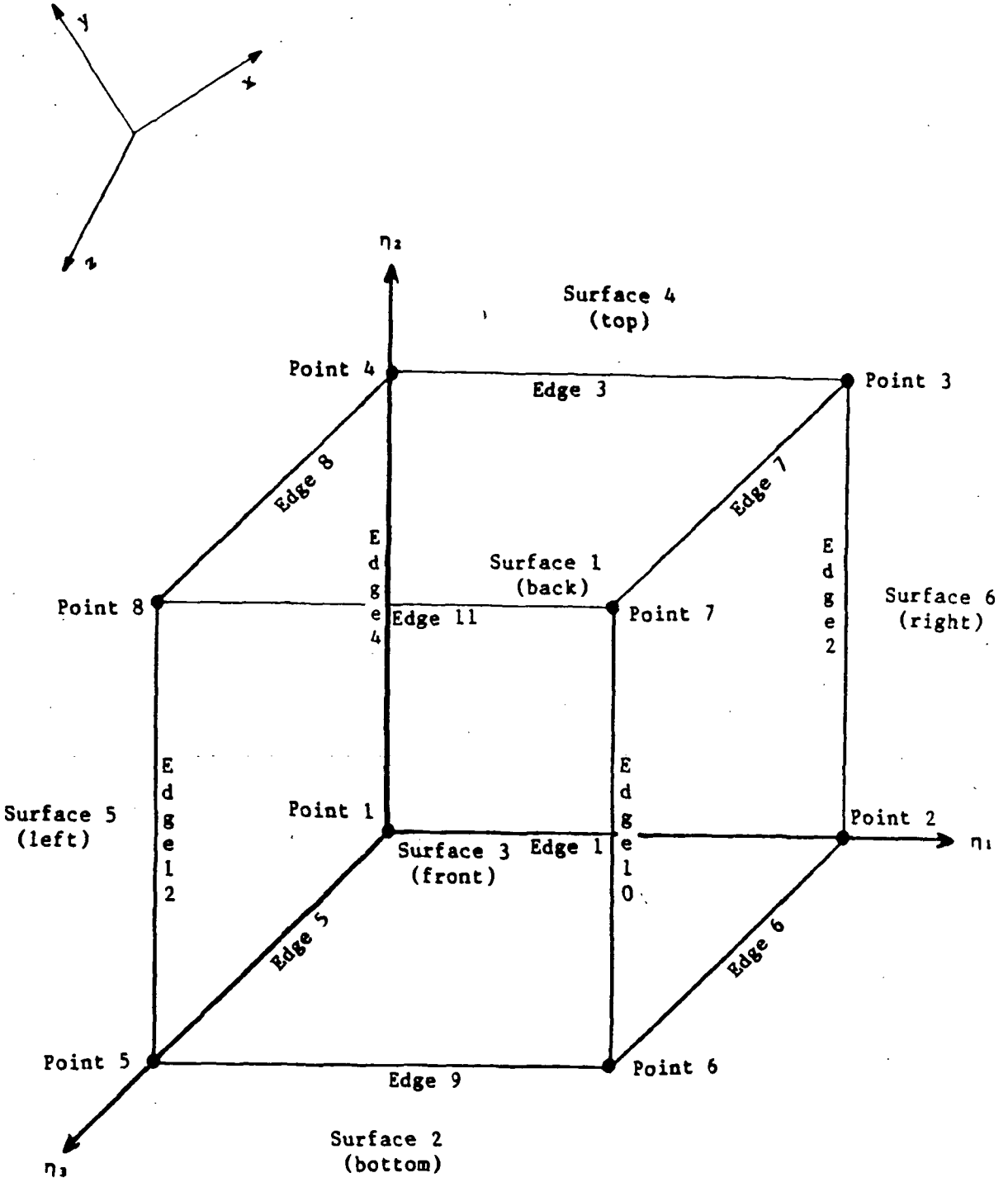


Fig. A-1 General Hexahedral: Numbering of Points, Edges, and Surfaces

DEFINITION OF TERMINOLOGY

Edge	An edge consists of from one to five segments. Four edges are used to describe a 2D section. Twelve edges are used to describe a 3D section.
Map	The geometry maps a point from η space into real space. When describing a surface mapping we could say map = 2 refers to a planar η space surface being mapped onto a cylindrical surface in real space.
Node	At each intersection of η coordinates a node is generated by the program forming the grid which will describe the flowfield domain.
Point	The corners of a section are called "points." The location and initial flow directions are input for each point. There are eight points.
Section	A zone may be further subdivided into sections. Each zone consists of at least one section. A zone may only be subdivided into sections in the direction in which the grid is generated (marching direction). Points, edges or a surface in common with a previous section within a zone are not reinput. Sectioning is done primarily to reduce the amount of input. Each section is described using points, edges, and surfaces.
Segment	An edge is subdivided into as many as five segments. A segment may be a straight line, a circular arc, a helical coil, a trigonometic function of angle or length, a cubic spline, or user defined.
Surface	A three-dimensional section will consist of six surfaces which form a generalized hexahedron. A surface may be planar, cylindrical, an edge of revolution, or user defined.
Zone	The flowfield domain may be subdivided into zones. Zones are generated independently and are the fundamental building block. Each zone contains its own η coordinate system.
η_1, η_2, η_3	Localized coordinate directions within a zone. These coordinates describe a cube in η space. $0 \leq \eta_I \leq 1$

SUMMARY OF CARDS

Card Type	Parameter List/Format
Flowfield Parameters	
1	ITITLE(I), I=1,80 (20A4)
2	NZONE, MAPTEN, INCHES (8I5)
3	IWRTI, IWRTC, IWRTN (3I5)
Zone Parameter	
4	NSECT (I5)
Section Parameters	
5	MAPEDGE(I), I=1,12 (12I5) or (3(4I10))
6	MAPSIDE(I), I=1,6 (6I5)
7	MARCH (I5)
8	(NMBRNDS(I), I=1,3), (ISTRTCH(I), I=1,3) (6I5)
9	STRETCH(I), I=1,3 (3E10.4)
10	[(COEFE(I,K,J), I=1,8), K=1,5], J=1,4(2D) OR 12(3D) (8E10.4)
11	[COEFS(I,J),I=1,8], J=1,6 (8E10.4)
12	(POINT(I,J), 8=15),J=1, 8 (8E10.4)
13	[(SEGMAX(I,K,J), I=1,5), ETAMAX(K,J), K=1,4] J=1, 12 (6E10.4)

CARD AND VARIABLE DESCRIPTIONS

Flowfield Parameters

CARD TYPE 1 Problem Identification Label Format(20A4)

ITITLE Alphanumeric information used for identifying the flowfield geometry. Columns 1-80 are read and printed only.

CARD TYPE 2 Problem Option Controls Flags Format(5I5)

NZONE The number of zones into which the flowfield geometry is divided. The maximum number of zones is 99.

MAPTEN This option determines the maximum number of segments which will be input per edge.

= 0 Five segments per edge Format(12I5)
= 1 Ten segments per edge Format(3(4I10))

INCHES This option specifies the dimensions of the coordinates being input. The output data will be written in feet for compatibility with the INTEGRATION program.

= 0 Dimensions in feet
= 1 Dimensions in inches

CARD TYPE 3 Output Print Options Format(3I5)

IWRTI This option may be used when either grid plots or nodal printout are insufficient to determine the cause of an error in the geometry. Intermediate debug printout may be generated in subroutines EDGE and SURFACE. Subroutine EDGE prints the following information: node number, edge number, position on the edge, η value at this position on the edge, a ratio which gives the relative location on a segment, and the maximum η value for this segment of the edge Subroutine SURFACE prints the following information: node number, surface number, and position on the surface. It also prints the following information for an edge of revolution: node number, surface number, position on the edge of revolution to be revolved, a vector tangent to the edge at this position, edge number, the distance along the axis of revolution from the origin of the relative coordinate system to a perpendicular from the axis to the point on the revolved edge, and the radius along this perpendicular from the axis to the revolved point on the edge. IWRTI should be chosen carefully to restrict the amount of printout.

= 0 No debug print
= N > 0 Intermediate error printout for every N^{th} node
= N > Total number of nodes - Prints the number of nodes stored plane.

IWRTC This option controls the printout of connectivities for all nodes specified for MATING. Incorrect connectivities result in various types of errors in the INTEGRATION program and a person familiar with it should be consulted if this type of error is suspected.

= 0 Do not print mated connectivities
= 1 Print mated connectivities

IWRTN This option controls the amount of nodal printout for a run. The nodal information will be printed will be node number and position. For large problems with many thousands of nodes, IWRTN should be chosen carefully to restrict the amount of paper produced. A plot of the grid is usually more instructive than a massive nodal printout for locating errors in the code.

= 0 No printout
= N > 0 Print nodal point information every N^{th} node

Zone Parameter

CARD TYPE 4 Number of Sections Format(I5)

NSECT The number of sections in this zone. A zone may only be sectioned in the η direction which corresponds to MARCH. There is no limit to the number of sections within a zone.

Section Parameters

CARD TYPE 5 Edge Shape Function Indicators Format(12I5) or 3(I10)

MAPEDGE(I), I=1, 12

These are packed integer flags that specify which edge shape functions will be used for the current section. The edges are input in numerical order. The edge numbers are defined according to Fig. A-1. The user should study this figure before inputting the geometry.

Each of the edges may consist of up to ten segments with each of these segments having its own shape function. The value of MAPEDGE(I) can consist of up to ten integers packed into one word MAPEDGE(I). MAPTEN specifies the maximum number of segments per edge. The edge shape function indicators for each segment are input in chronological order of increasing η for each edge with the final packed integer being right adjusted. For example, if MAPEDGE(4) = 112, then edge 4 consists of three segments: the first segment is type 1; the second segment is type 1; and the third segment is type 2. If only one segment describes an edge, then only <u>one</u> indicator is used, right adjusted.

A library of edge shape functions indicators for the HGM GEOMETRY program follow. If any edge shape function other than a linear segment is specified, then edge coefficients (COEFE(I)) must be input. Card type 1 is used to define the analytical function describing a segment.

= 1	Linear segment	
= 2	Circular arc	(input COEFE(I))
= 3	Edge of revolution	(input COEFE(I))
= 4	Special segment	(input COEFE(I))
= 5	Special segment	(input COEFE(I))

CARD TYPE 6 Surface Shape Function Indicators Format(6I5)

<u>MAPSIDE(I),I=1,6</u>

These are integer flags that specify which surface shape functions will be used for the current section. These flags are input <u>only</u> for three-dimensional problems since two-dimensional geometries are defined completely by the edge functions. The surfaces are input in numerical order. The surface numbers are defined in Fig. A-1. The user should study this figure before inputting the geometry. An edge of revolution requires the input of surface coefficients (COEFS(I)) on card type 15 to define a relative origin on the axis, the axis of revolution, and the direction of revolution.

= 1	Planar surface	
= 2	Cylindrical surface	
= 3	Special surface	(user defined)
= 4	Edge of revolution	(input COEFS(I))
= 5	Holes in bowl surface	
= 6	Hole side of duct surface	
= 7	Duct outer surface	

CARD TYPE 7 Node Numbering Sequence Specs Format(6I5)

MARCH(I5) The value of MARCH determines the node generation and hence the node numbering sequence. (default = 1)

The numbering sequence corresponding to follows.

= 1 η_3, η_2, η_1 (default)
= 2 η_1, η_3, η_2
= 3 η_2, η_1, η_3

CARD TYPE 8 Node Distribution Parameters Format(6I5)

NMBRNDS(I),I=1 3

NMBRNDS(I) is the number of nodes in the η_I direction for the current section. The limit is 200 nodes in any coordinate direction. This may be changed in the program by respecifying the ETAS(3,200) array.

ISTRTCH(I),I=1 3

This option gives the user control over the node distribution in each of the coordinate directions.

= 0 Uniform spacing
= 1 Input actual η_I values for NMBRNDS(I) nodes (input ETAS(I))
= 2 Decrease spacing in η_I direction. Input a stretching factor greater than 0.0 in STRETCH(I).
= 3 Increase spacing in η_I direction. Input a stretching factor greater than 0.0 in STRETCH(I).
= 4 Double stretching. Input a stretching factor greater than 0.0 in STRETCH(I). Use an odd number of nodes.
= 5 Decrease spacing in η_I direction. Input minimum grid spacing as a percentage of the total length in STRETCH(I).
= 6 Increase spacing in η_I direction. Input minimum grid spacing as a percentage of the total length in STRETCH(I).
= 7 Double stretching. Input minimum grid spacing as a percentage of the total length in STRETCH(I). Use an odd number of nodes.

If ISTRTCH(I) = 1, input a set of cards type 13 for each η direction to be input.
If ISTRTCH(I) $\geq$ 2, input card type 12.

CARD TYPE 9 Option for Stretching Function (input when ISTRTCH(I) $\geq$ 2) Format(3E10.4)

STRETCH(I),I=1, 3

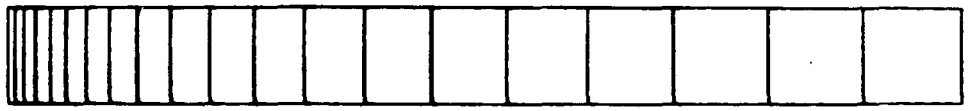
This parameter is input for each coordinate direction designated for stretching by ISTRTCH(I) $\geq$ 2.

Example: Several stretching functions will be demonstrated using 21 points for comparison. Note, that total length = 10.0 for ISTRTCH = 6 and 7.

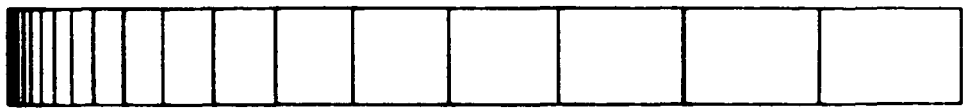
ISTRTCH = 3

STRETCH = 2.0

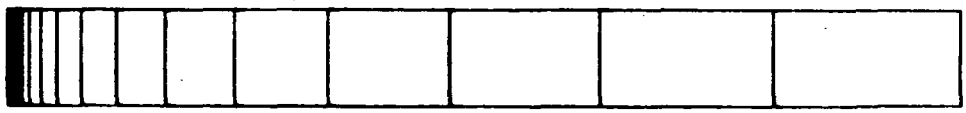
STRETCH = 4.0



STRETCH = 6.0



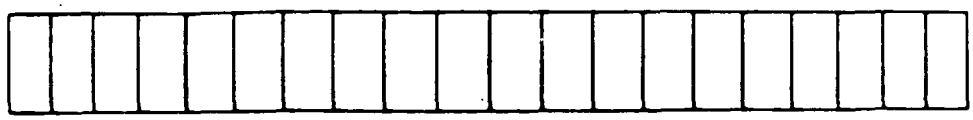
STRETCH = 8.0



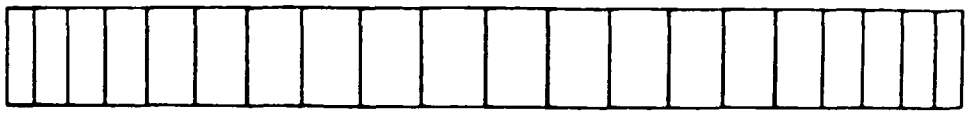
STRETCH = 10.0

ISTRTCH = 4

STRETCH = 2.0



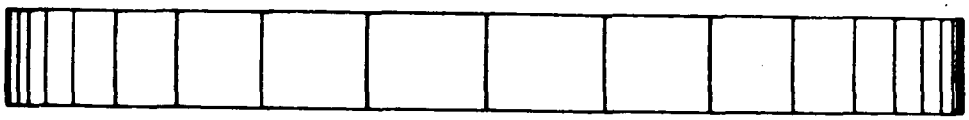
STRETCH = 4.0



STRETCH = 6.0

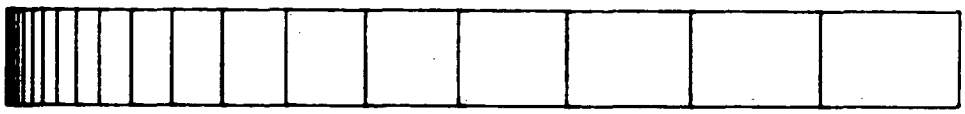
STRETCH = 8.0

STRETCH = 10.0

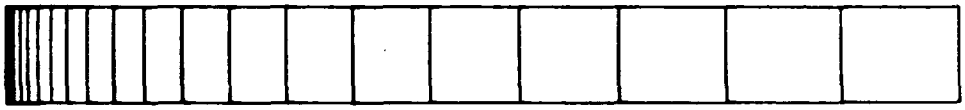


ISTRTCH = 6

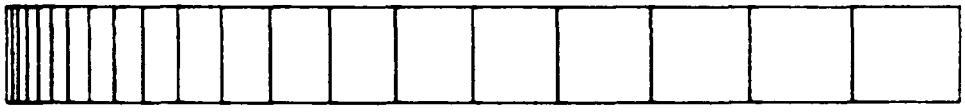
STRETCH = .002



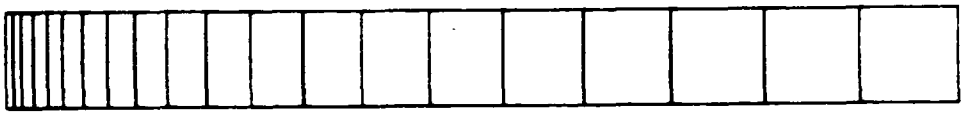
STRETCH = .004

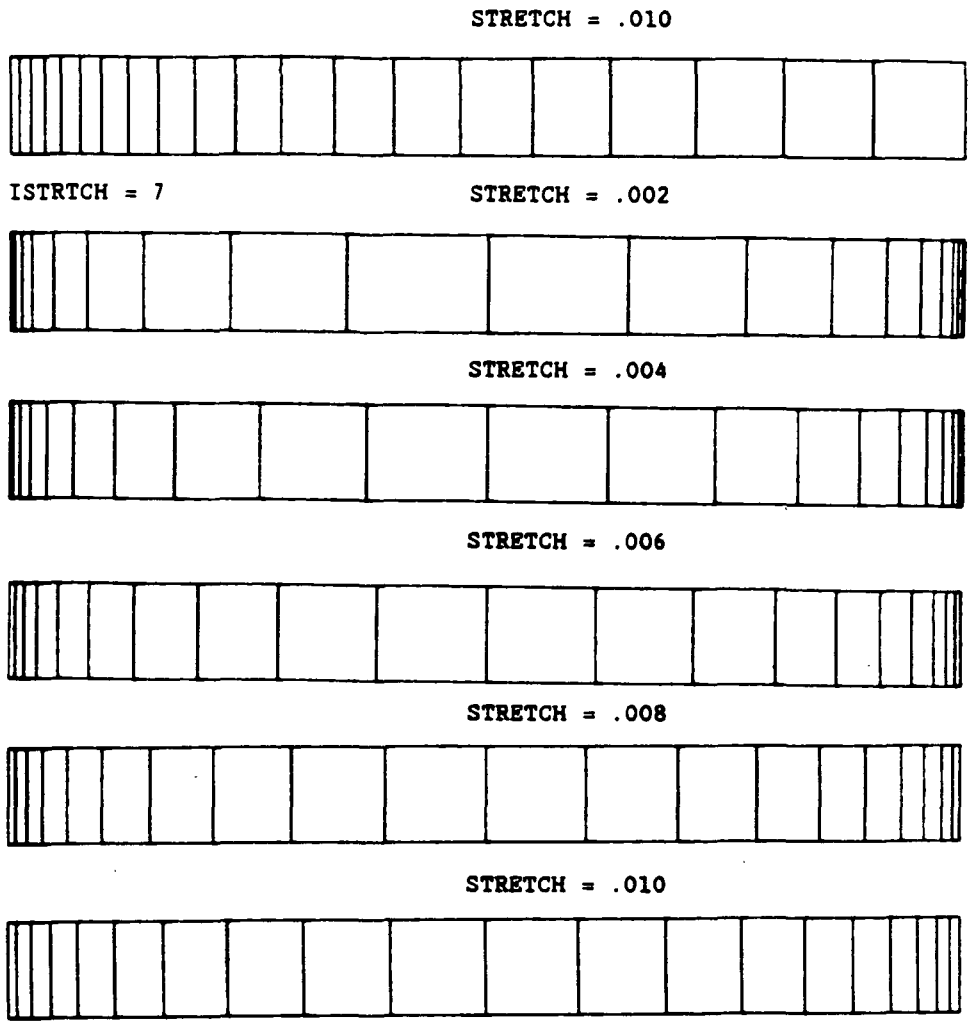


STRETCH = .006



STRETCH = .008





CARD TYPE 14 Coefficients for Edge Shape Functions Format(8E10.4)
(input for each segment in MAPEDGE(I) > 1)

COEFE(I),I=1,8

These coefficients are used to describe the edge shape functions for each segment of the current section. The coefficients for each segment are input on separate cards in the same order as the indicators on card type 5.

Map	Type	Input Parameters
2	Circular Arc	COEFE(I), I=1,3 are the x, y, and z coordinates of the center of the arc.
3	Edge of Revolution	COEFE(I), I=1,3 are coordinates of the center of the arc. COEFE(I), I=4,6 are components of the unit vector along the axis. Direction according to right hand rule when a vector is revolved from point 1 to point 2.

CARD TYPE 15 Coefficients for Surface Shape Functions Format(8E10.4)
(input when MAPSIDE(I) = 4 and IDIM = 3)

COEFS(I),I=1,8

These are the coefficients defining the surface shape functions for each surface formed by an edge of revolution in the current section. Each surface which has MAPSIDE(I) = 4 on card type 8 is input on a separate card in the same order as they occur on card type 8.

Map	Type	Input Parameters
4	Surface of Revolution	Surface formed by revolving an edge about an axis. COEFS(I), I=1,3 a point on the axis of revolution which becomes the origin of a local coordinate system. This point must lie outside of the projection of the edge onto the axis of revolution. COEFS(I), I=4,6 are components of the unit vector along the axis of revolution in the direction of increasing n. COEFS(7) indicates the η_I direction in which the edge will revolve.

CARD TYPE 16 Coordinates of Points and flow direction Format(5E10.4)

POINT(I,J), J = Point number

These parameters are the coordinates and flow direction at each corner of a general hexahedral(3D). Figure A-1 shows this configuration with the points numbered from 1 to 8. There are eight cards of type 16 to be input.

POINT(1,J) - the x coordinate of point J

POINT(2,J) - the y coordinate of point J

POINT(3,J) - the z coordinate of point J

POINT(4,J) - the flow angle θ at point J

POINT(5,J) - the flow angle Φ at point J

Important Note: All cards type 12 are not input consecutively. They are grouped with cards type 13. See Table A-1 for the exact sequence of card types 11 and 13.

CARD TYPE 17 Segment Extremals for Edges Format(6E10.4)

SEGMAX(I,K,J), K = Segment Number, J = Edge Number

Each edge may be segmented up to five times. Therefore, cards type 13 are repeated for each successive segment of edge J. Each segment must be input on a separate card type 13. The extremal for the final segment of an edge is not to be input since this point is already defined by the POINT(I) input. The number of cards type 13 for each edge will thus be one less than the number of segments on that edge. In particular, if an edge consists of only one segment, no cards of type 13 are input for that edge.

See Table 1 for the input order of card types 12 and 13. Each POINT(I) is input on a single card, followed by up to five cards containing the extremals.

SEGMAX(1,K,J) - The extremal x coordinate for the K^{th} segment of edge J.
SEGMAX(2,K,J) - The extremal y coordinate for the K^{th} segment of edge J.
SEGMAX(3,K,J) - The extremal z coordinate for the K^{th} segment of edge J.
SEGMAX(4,K,J) - The extremal flow angle θ for the K^{th} segment of edge J.
SEGMAX(5,K,J) - The extremal flow angle Φ for the K^{th} segment of edge J.

ETAMAX(K,J), K= Segment Number, J= Edge Number

The maximum value of the η_I coordinate on the K^{th} segment of edge J. Input a negative value when defining a fold line.

Table A-1 INPUT SEQUENCE FOR POINTS AND EXTREMALS OF SEGMENTS: CARDS TYPE 16 AND 17

Order	Card Type	Description
1	12	Point 1
2	13	Extremals for each segment of Edge 1
3	12	Point 2
4	13	Extremals for each segment of Edge 2
5	12	Point 3
6	13	Extremals for each segment of Edge 3
7	12	Point 4
8	13	Extremals for each segment of Edge 4
9	12	Point 5
10	13	Extremals for each segment of Edge 5
11	12	Point 6
12	13	Extremals for each segment of Edge 6
13	12	Point 7
14	13	Extremals for each segment of Edge 7
15	12	Point 8
16	13	Extremals for each segment of Edge 8
17	13	Extremals for each segment of Edge 9
18	13	Extremals for each segment of Edge 10
19	13	Extremals for each segment of Edge 11
20	13	Extremals for each segment of Edge 12

Notes on Input Sequence of Cards Type 4 Through 13

- Card types 5-13 are repeated for each section in a zone. Sections can be added only in the direction of MARCH.
- Card type 4 is repeated for each zone.
- Cards type 9-13 are not re-input for points, edges and surfaces in common with the preceding section. Since these have already been input, the code transfers the entries from one section to another.
- The first section of each new zone, however, must be input since zones are considered to be independent.

Appendix B
HGM GRID CODE INPUT LISTING

Appendix B

```
 ZONE 1 (TAD) FOR 3-DUCT HGM GEOMETRY
    1    3    1    1
10000    0    0
    1                                                          ZONE 1
      1311          1       1311          1
         3          3          3          3
      1311          1       1311          1
    1    4    1    4    4    4
    3
   94   21   72    0    4    0
 0.0        8.0        0.0
 1.35       0.0       -6.0        0.0       -1.0       0.0
 1.35       0.0       -5.875      0.0       -1.0       0.0
 3.31       0.0        0.0        1.0        0.0       0.0
 6.07       0.0        0.0        1.0        0.0       0.0
 6.07       0.0        0.0        1.0        0.0       0.0
 3.31       0.0        0.0        1.0        0.0       0.0
 1.35       0.0        6.0        0.0        1.0       0.0
 1.35       0.0        5.875      0.0        1.0       0.0
 6.07       0.0        0.0        1.0        0.0       0.0        3.0
 6.07       0.0        0.0        1.0        0.0       0.0        3.0
 6.07       0.0        0.0        1.0        0.0       0.0        3.0
 6.07       0.0        0.0        1.0        0.0       0.0        3.0
 3.31       0.0       -5.55
 1.35       0.0       -5.55       18.0
 1.35       0.0       -6.45       56.0
 4.02       0.0       -6.5        81.0
 6.07       0.0       -6.5
 6.07       0.0       -7.434
 1.35       0.0       -4.525      18.0
 1.35       0.0       -7.225      55.0
 4.02       0.0       -7.434      81.0
 3.31       0.0       -4.525
 3.31       0.0        5.55
 6.07       0.0        6.5
 6.07       0.0        7.434
 3.31       0.0        4.525
 1.35       0.0        5.55       18.0
 1.35       0.0        6.45       56.0
 4.02       0.0        6.5        81.0
 1.35       0.0        4.525      18.0
 1.35       0.0        7.225      55.0
 4.02       0.0        7.434      81.0
```

```
 ZONE 2 (1/2 BOWL) FOR 3-DUCT HGM GEOMETRY
    1    3    1    1
10000    0    0
    1                                                          ZONE 2
 111111111          1     121111          1
         3          3          3          3
 111111111          1     121111          1
    1    4    1    4    4    4
    3
```

```
   59    21    36     0     4     0
 0.0         8.0         0.0
 9.445       0.0         0.0
 0.0         0.0         0.0         1.0         0.0         0.0
 0.0         0.0         0.0         1.0         0.0         0.0
 0.0         0.0         0.0         1.0         0.0         0.0
 0.0         0.0         0.0         1.0         0.0         0.0
 9.445       0.0         0.0
 0.0         0.0         0.0         1.0         0.0         0.0         3.0
 0.0         0.0         0.0         1.0         0.0         0.0         3.0
 0.0         0.0         0.0         1.0         0.0         0.0         3.0
 0.0         0.0         0.0         1.0         0.0         0.0         3.0
 6.07        0.0        -6.5
 6.8015659   0.0        -6.5         4.0
 7.8         0.0        -6.5        11.0
 9.1         0.0        -6.62       21.0
 9.78        0.0        -6.62       26.0
13.555       0.0        -6.37       54.0
13.8395195   0.0        -6.2211618
13.8395195   0.0        -6.5734119
 6.8015659   0.0        -7.434       4.0
13.39        0.0        -6.8329404  53.0
 6.07        0.0        -7.434
 6.07        6.5         0.0
13.8395195   6.2211618   0.0
13.8395195   6.5734119   0.0
 6.07        7.434       0.0
 6.8015659   6.5         0.0         4.0
 7.8         6.5         0.0        11.0
 9.1         6.62        0.0        21.0
 9.78        6.62        0.0        26.0
13.555       6.37        0.0        54.0
 6.8015659   7.434       0.0         4.0
13.39        6.8329404   0.0        53.0
```

```
 ZONE 3 (1/2 BOWL) FOR 3-DUCT HGM GEOMETRY
    1     3     1     1
10000     0     0
    1                                                          ZONE 3
 111111111           1     121111          1
         3           3          3          3
    111111           1        121          1
    1     4     1     5     4     4
    3     0     1
        4.0         20.0        46.0
       13.0         41.0        58.0
        3.0          4.0         4.0
        3.0          4.0         4.0        3.0
   59    21    72     0     4     0
 0.0         8.0         0.0
 9.445       0.0         0.0
 0.0         0.0         0.0         1.0         0.0         0.0
 0.0         0.0         0.0         1.0         0.0         0.0
 0.0         0.0         0.0         1.0         0.0         0.0
 0.0         0.0         0.0         1.0         0.0         0.0
 9.445       0.0         0.0
```

```
 0.0         0.0         0.0         1.0         0.0         0.0         3.0
 0.0         0.0         0.0         1.0         0.0         0.0         3.0
 0.0         0.0         0.0         1.0         0.0         0.0         3.0
 0.0         0.0         0.0         1.0         0.0         0.0         3.0
 6.07        6.5         0.0
 6.8015659   6.5         0.0         3.0
 7.8         6.5         0.0         11.0
 9.1         6.62        0.0         21.0
 9.78        6.62        0.0         26.0
13.555       6.37        0.0         55.0
13.839518    6.221162    0.0
13.839518    6.573411    0.0
 6.8015659   7.434       0.0          4.0
13.39        6.8329404   0.0         53.0
 6.07        7.434       0.0
 6.07        0.0         6.5
13.839518    0.0         6.221162
13.839518    0.0         6.573411
 6.07        0.0         7.434
 6.8015659   0.0         6.5          4.0
 7.8         0.0         6.5         11.0
 9.1         0.0         6.62        21.0
 9.78        0.0         6.62        26.0
13.555       0.0         6.37        55.0
 6.8015659   0.0         7.434        4.0
13.39        0.0         6.8329404   53.0
```

```
 ZONE 4 (TOP T-D) FOR 3-DUCT HGM GEOMETRY
    1     3     1     1
10000     0     0
    3                                                              ZONE 4
          4           5           3           5
          4           4           3           3
          4           5           3           5
    7     6     7     1     7     7
    2
   27    20    29     4     0     4
 7.0         0.0         7.0
 0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0
 9.445       5.564       0.0        -0.25506    -0.092215   -0.96252
 0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0
 9.445       5.564       0.0        -0.25506    -0.092215   -0.96252
 9.445       5.564       0.0         0.25506     0.092215    0.96252
 0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0
 9.445       5.564       0.0         0.25506     0.092215    0.96252
 0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0         0.0
 8.691001    7.428427    2.549918
```

```
11.362797   7.428428   1.841919
12.66737    7.79672    7.24362
10.23170    7.79672    7.88905
 9.568123   4.613040   6.399753
12.403425   4.49657    5.768945
12.73830    4.80664    7.51127
10.17804    4.92167    8.17869
         3          1          3          1
         3          3          3          3
         3          1          3          1
    4    1    4     1    4     4
    2
   27   13   29     4    0     4
 7.0         0.0        7.0
 9.445       5.564      0.0       -0.25506  -0.092215 -0.96252
 9.445       5.564      0.0       -0.25506  -0.092215 -0.96252
 9.445       5.564      0.0        0.25506   0.092215  0.96252
 9.445       5.564      0.0        0.25506   0.092215  0.96252
 9.445       5.564      0.0       -0.25506  -0.092215 -0.96252   1.0
 9.445       5.564      0.0        0.25506   0.092215  0.96252   1.0
 9.445       5.564      0.0        0.25506   0.092215  0.96252   3.0
 9.445       5.564      0.0       -0.25506  -0.092215 -0.96252   3.0
14.18025    8.61610   11.69913
11.12323    8.61610   12.50921
14.26927    4.86326   12.03505
11.05588    5.00762   12.87274
         3          1          3          1
         3          3          3          3
         3          1          3          1
    4    1    4     1    4     4
    2
   27   13   29     4    0     4
 7.0         0.0        7.0
 9.445       5.564      0.0       -0.25506  -0.092215 -0.96252
 9.445       5.564      0.0       -0.25506  -0.092215 -0.96252
 9.445       5.564      0.0        0.25506   0.092215  0.96252
 9.445       5.564      0.0        0.25506   0.092215  0.96252
 9.445       5.564      0.0       -0.25506  -0.092215 -0.96252   1.0
 9.445       5.564      0.0        0.25506   0.092215  0.96252   1.0
 9.445       5.564      0.0        0.25506   0.092215  0.96252   3.0
 9.445       5.564      0.0       -0.25506  -0.092215 -0.96252   3.0
15.51900    9.10020   16.75236
12.46198    9.10020   17.56244
15.60802    5.34736   17.08828
12.39463    5.49172   17.92597


ZONE 5 (MID T-D) FOR 3-DUCT HGM GEOMETRY
    1    3    1    1
    0    0    0
    3                                                        ZONE 5
         4          5          3          5
         4          4          3          3
         4          5          1          5
    7    6    1    1    7    7
    2
   27   19   16    4    0    3
```

```
 7.0         0.0         5.0
 0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0
 7.32766     0.0         0.37334    -0.40139     0.0        -0.91591
 0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0
 7.32766     0.0         0.37334    -0.40139     0.0        -0.91591
 7.32766     0.0         0.37334     0.40139     0.0         0.91591
 0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0         0.0
 0.0         0.0         0.0         0.0         0.0         0.0         0.0
 9.040005    1.695449    7.695033
11.889400    1.695342    7.307725
12.3459      1.5014      8.6854
12.5633      1.5014      9.6966
 8.233353    0.0         7.796413
12.667614    0.0         7.201867
12.98742     0.0         8.40500
 9.39706     0.0         9.97844
         3           1           3           1
         3           3           3           3
         1           1           1           1
    4    1     1     1     4     4
    2
   27   25    16     4     0     3
 7.0         0.0         5.0
 7.32766     0.0         0.37334    -0.40139     0.0        -0.91591
 7.32766     0.0         0.37334     0.40139     0.0         0.91591
 7.32766     0.0         0.37334     0.40139     0.0         0.91591
 7.32766     0.0         0.37334    -0.40139     0.0        -0.91591    1.0
 7.32766     0.0         0.37334     0.40139     0.0         0.91591    3.0
 7.32766     0.0         0.37334    -0.40139     0.0        -0.91591    3.0
14.7473      1.8845     13.3654
11.8507      1.8845     14.6346
15.55268     0.0        13.01283
11.04640     0.0        14.98767
         3           1           3           1
         3           3           3           3
         1           1           1           1
    4    1     1     1     4     4
    2
   27    4    16     4     0     3
 7.0         0.0         5.0
 7.32766     0.0         0.37334    -0.40139     0.0        -0.91591
 7.32766     0.0         0.37334     0.40139     0.0         0.91591
 7.32766     0.0         0.37334     0.40139     0.0         0.91591
 7.32766     0.0         0.37334    -0.40139     0.0        -0.91591    1.0
 7.32766     0.0         0.37334     0.40139     0.0         0.91591    3.0
 7.32766     0.0         0.37334    -0.40139     0.0        -0.91591    3.0
14.9963      1.8845     13.9344
12.0997      1.8845     15.2036
15.80235     0.0        13.58253
11.29607     0.0        15.55737
```

Appendix C
HGM GRID CODE LISTING

Appendix C

```
C*************************************************************************
C*****MAIN***************************************************************
C*************************************************************************
C
C       PROGRAM HGMGEOM(INPUT,OUTPUT,FILE19,FILE20,TAPE5,TAPE6=OUTPUT,
C      &               TAPE19=FILE19,TAPE20=FILE20)
C
      PROGRAM HGMGEOM
C-----------------------------------------------------------------------
C   DEVELOPED BY THE COMPUTATIONAL MECHANICS SECTION
C   LOCKHEED ENGINEERING CENTER, HUNTSVILLE, ALABAMA.
C                                        L.A. NICHOLSON
C
C   TAPE19 GEOMETRY SCRATCH FILE
C   TAPE20 GEOMETRY DATA
C-----------------------------------------------------------------------
C
      COMMON /COUNTER/ NODESAV,NODETOT,NBNODES,NPLANE
      COMMON /INITA/   IDIM,MAPTEN,INCHES
      COMMON /ZONING/  ISECT,NSECT,IZONE,NMBRNDS(3)
      COMMON /UNITS/   NU5,NU6,NU19,NU20,NU21
C
C---INITIALIZE PROGRAM
C
      CALL INITIAL(NZONE)
C
C-----------------------------------------------------------------------
C   ZONE
C-----------------------------------------------------------------------
      DO 300 IZONE=1,NZONE
C
C
      READ(NU5,1000) NSECT
C
C-----------------------------------------------------------------------
C   SECTION
C-----------------------------------------------------------------------
C
      DO 200 ISECT=1,NSECT
C
C---READ INPUT FOR EACH SECTION
C
      CALL INPUT
C
C---GENERATE GRID FOR EACH SECTION
C
  200 CALL GRID
C
      CALL REWRITE
C
  300 CONTINUE
C
C
      STOP
C
C---FORMAT STATEMENTS
C
 1000 FORMAT(I5)
C
      END
```

```
C
C*************************************************************************
C*****INPUT**************************************************************
C*************************************************************************
C
      SUBROUTINE INITIAL(NZONE)
C
C
      COMMON /COEFF/    COEFE(8,10,12),COEFS(8,6),NMBRSEG(12)
      COMMON /COUNTER/  NODESAV,NODETOT,NBNODES,NPLANE
      COMMON /HEADER/   ITITLE(20),LINE
      COMMON /INITA/    IDIM,MAPTEN,INCHES
      COMMON /INITB/    IBOX(4,6),IRAY(4,3),NRAY,IPT1(12),IPT2(12)
      COMMON /INITC/    PI,RADDEG
      COMMON /IWRITE/   IWRTC,IWRTI,IWRTN
      COMMON /NODNMBR/  NODENUM(10000),TOL(3)
      COMMON /UNITS/    NU5,NU6,NU19,NU20,NU21
C
      DIMENSION NAME(20)
C
      DATA PI       /3.141592654/
      DATA RADDEG /57.29577951/
      DATA IPT1 /1,2,4,1,1,2,3,4,5,6,8,5/
      DATA IPT2 /2,3,3,4,5,6,7,8,6,7,7,8/
      DATA IRAY /1,3,9,11,4,2,12,10,5,6,8,7/
      DATA IBOX /4,1,2,3,5,1,6,9,12,9,10,11,8,3,7,11,5,4,8,12,6,2,7,10/
C-------------------------------------------------------------------------
C            1   2   3   4   5   6   7   8   9  10  11  12
C           ---------------------------------------------
C    IPT1 =  1   2   4   1   1   2   3   4   5   6   8   5
C    IPT2 =  2   3   3   4   5   6   7   8   6   7   7   8
C
C    IRAY =    1   4   5
C              3   2   6
C              9  12   8
C             11  10   7
C
C    IBOX =    4   5  12   8   5   6
C              1   1   9   3   4   2
C              2   6  10   7   8   7
C              3   9  11  11  12  10
C
C-------------------------------------------------------------------------
C---DYNAMIC DIMENSIONER INPUT
C
C
C---INITIALIZE
C
              NU5   = 5
              NU6   = 6
              NU19  = 19
              NU20  = 20
              NU21  = 21
            NPLANE  = 0
           NBNODES  = 0
           NINODES  = 0
           NODETOT  = 0
           NODESAV  = 0
C
      DO 40 I=1,8
```

```
C
      DO 30 J=1,6
   30   COEFS(I,J) = 0.0
C
      DO 40 J=1,10
      DO 40 K=1,12
   40 COEFE(I,J,K) = 0.0
C
C
C---PRINT HEADER
C
      WRITE(NU6,1120)
C-----------------------------------------------------------------------
C   NZONE     = NUMBER OF ZONES TO BE USED TO GENERATE GRID
C   IDIM      = 2  FOR 2D
C             = 3  FOR 3D
C   MATING    = 0 NO ZONE MATING
C   IWRTI     = 0 NO INTERMEDIATE PRINT
C   IWRTC     = 0 NO CONNECTIVITY PRINTOUT
C   IWRTN     = 0 PRINT ONLY BOUNDARY NODES
C             = N PRINT EVERY NTH INTERIOR NODE
C   MAPTEN    = 0 FIVE SEGMENTS PER EDGE
C             = 1 TEN SEGMENTS PER EDGE
C   INCHES    = 0 COORDINATES INPUT IN FEET
C             = 1 COORDINATES INPUT IN INCHES
C-----------------------------------------------------------------------
      READ(NU5,1000) ITITLE
      READ(NU5,1010) NZONE,IDIM,
     &               MAPTEN,INCHES
C
C
      READ(NU5,1010) IWRTI,IWRTC,IWRTN
C
C
                       NRAY = 2*(IDIM - 1)
C
                     WRITE(NU6,1130) ITITLE
                     WRITE(NU6,1140) NZONE,IDIM,
     &                                MAPTEN,INCHES
C
      REWIND(NU20)
C
C---DRAW PICTURE
C
      CALL PICTURE(IDIM)
C
      RETURN
C
C---FORMAT STATEMENTS
C
 1000 FORMAT(20A4)
 1010 FORMAT(16I5)
 1120 FORMAT(1H1
     3    / 40X,34H               LOCKHEED-HUNTSVILLE
     4    / 40X,34H                VAX 11/785 VERSION
     5    / 40X,32H                   HGM GEOMETRY )
 1130 FORMAT( // 5X,20A4 )
 1140 FORMAT(// 21H GEOMETRY PARAMETERS:
     1       // 45H     NZONE      IDIM
     2       MAPTEN    INCHES /, 8I10 )
```

```
C
      END
C
C*************************************************************************
C*****INPUT**************************************************************
C*************************************************************************
C
      SUBROUTINE INPUT
C
C
      COMMON /COEFF/   COEFE(8,10,12),COEFS(8,6),NMBRSEG(12)
      COMMON /COUNTER/ NODESAV,NODETOT,NBNODES,NPLANE
      COMMON /HEADER/  ITITLE(20),LINE
      COMMON /HOTGAS/  IHGM,ETA11,ETA12,ETA13,ETA31,ETA32,ETA33,
     &                 STR11,STR12,STR13,STR31,STR32,STR33,STR34
      COMMON /INITA/   IDIM,MAPTEN,INCHES
      COMMON /INITB/   IBOX(4,6),IRAY(4,3),NRAY,IPT1(12),IPT2(12)
      COMMON /INPUTA/  EDGE(6,12),POINT(6,8),SIDE(6,6)
      COMMON /IWRITE/  IWRTC,IWRTI,IWRTN
      COMMON /MAPING/  MAPSIDE(6),MAPSEG(10,12)
      COMMON /MARCHS/  MARCH,INDEX(3)
      COMMON /MAXIMUM/ ETAMAX(10,12),SEGMAX(6,10,12)
      COMMON /SPACING/ ISTRTCH(3),STRETCH(3),ETAS(3,200),ETA(3),DETA(3)
      COMMON /SPLIT/   NSPLIT,ISURF(10),IBC(10),
     &                 ISTART1(10),INC1(10),ISTART2(10),INC2(10)
      COMMON /UNITS/   NU5,NU6,NU19,NU20,NU21
      COMMON /ZONING/  ISECT,NSECT,IZONE,NMBRNDS(3)
C
      DIMENSION IFLAGP(8),IFLAGE(12),IFLAGS(6)
      DIMENSION LSTRTCH(3),LBCINPT(6),LMAPS(6)
      DIMENSION MAPEDGE(12),STRTCH(3),LNMBRND(3)
C-------------------------------------------------------------------------
C   MAPEDGE(I)   INDICATES TYPE OF GEOMETRY FOR EDGE I
C                = 1 LINEAR
C                = 2 CIRCULAR ARC
C                = 3 EDGE OF REVOLUTION
C
C   MAPSIDE(I)   TYPE OF GEOMETRY FOR SURFACE I
C                = 1 FLAT SURFACE
C                = 4 EDGE OF REVOLUTION
C                = 5,6,7 SPECIAL SURFACE SYBROUTINE
C
C   MARCH        ETA DIRECTION IN WHICH COMPUTATION IS TO ADVANCE
C
C   NMBRNDS(I)   NUMBER OF NODES IN EACH ETA(I) DIRECTION
C
C   ISTRTCH(I)   =  0 NO STRETCHING IN ETA(I) DIRECTION
C                =  1 INPUT N VALUES OF ETA(I)
C                =  2 TIGHTEN GRID IN ETA(I) DIRECTION
C                =  3 LOOSEN GRID IN ETA(I) DIRECTION
C-------------------------------------------------------------------------
      WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
C
      IF(ISECT.EQ.1) GO TO 100
C-------------------------------------------------------------------------
C   SAVE INPUT PARAMETERS IN COMMON WITH PREVIOUS SECTION
C-------------------------------------------------------------------------
      DO 10 I=1,3
      LSTRTCH(I) = ISTRTCH(I)
   10 LNMBRND(I) = NMBRNDS(I)
```

```
C
      DO 20 I=1,6
   20   LMAPS(I) = MAPSIDE(I)
C-----------------------------------------------------------------------
C   READ INPUT PARAMETERS
C-----------------------------------------------------------------------
  100 CONTINUE
C
       READ(NU5,1030) MAPEDGE
C
       READ(NU5,1000) MAPSIDE
       READ(NU5,1000) MARCH,NSPLIT,IHGM
C
      IF(IHGM.EQ.1) THEN
C
                     READ(NU5,1020) ETA11,ETA12,ETA13
                     READ(NU5,1020) ETA31,ETA32,ETA33
                     READ(NU5,1020) STR11,STR12,STR13
                     READ(NU5,1020) STR31,STR32,STR33,STR34
C
                     END IF
C
C---READ INPUT FOR GENERALIZED BOUNDARY CONDITIONS
C
      READ(NU5,1000) NMBRNDS,ISTRTCH
C
C---PRINT OUT INPUT VARIABLES
C
                     WRITE(NU6,3000) MAPEDGE
                     WRITE(NU6,3010) MAPSIDE
                     WRITE(NU6,3020) MARCH
                     WRITE(NU6,3040) NMBRNDS,ISTRTCH
C
C---INITIALIZE FLAGS TO CHECK IF POINT, EDGE, OR SURFACE HAS BEEN INPUT
C
      DO 130 I=1,8
  130 IFLAGP(I) = 0
C
      DO 140 I=1,12
  140 IFLAGE(I) = 0
C
      DO 150 I=1,6
  150 IFLAGS(I) = 0
C
C---SET ORDER OF EXECUTION
C
      INDEX(1) = MARCH
      INDEX(3) = 3
C
      DO 160 I=2,IDIM
                           INDEX(I) = INDEX(I-1) + 1
  160 IF(INDEX(I).GT.IDIM) INDEX(I) = 1
C
C
C---INITIALIZE ETA
C
      ETA(1) = 0.0
      ETA(2) = 0.0
      ETA(3) = 0.0
C
```

```
      IF(ISECT.EQ.1 ) GO TO 300
C-------------------------------------------------------------------------
C   TRANSFER INPUT PARAMETERS IN COMMON WITH PREVIOUS SECTION
C-------------------------------------------------------------------------
C---TRANSFER POINTS
C
      DO 200 I=1,NRAY
                        L = IRAY(I,MARCH)
                   NEWPT = IPT1(L)
                  LASTPT = IPT2(L)
            IFLAGP(NEWPT) = 1
C
      DO 200 J=1,6
  200       POINT(J,NEWPT) = POINT(J,LASTPT)
C
C---DETERMINE WHICH TWO SURFACES ARE IN COMMON
C
C
                     NEWSRF = 10 + (MARCH - 6)*MARCH
                    LASTSRF =  9 + (MARCH - 7)*MARCH/2
C
C
C---NUMBER OF EDGES IN COMMON WITH PREVIOUS SECTION
C
                  MEDGES = 3*IDIM - 5
C
      DO 230 I=1,MEDGES
C
C---DETERMINE WHICH EDGES ARE IN COMMON WITH PREVIOUS SECTION
C
C
                     NEWEDG = IBOX(I, NEWSRF)
                    LASTEDG = IBOX(I,LASTSRF)
C
C
       IFLAGE(NEWEDG) = 1
C
C---TRANSFER NUMBER OF SEGMENTS
C
      NMBRSEG(NEWEDG) = NMBRSEG(LASTEDG)
C
C---TRANSFER SEGMENT MAPPING
C
      DO 210 J=1,NMBRSEG(LASTEDG)
C
      MAPSEG(J,NEWEDG) = MAPSEG(J,LASTEDG)
C
C---TRANSFER EDGE COEFFICIENTS
C
      DO 210 K=1,8
       COEFE(K,J, NEWEDG) = COEFE(K,J,LASTEDG)
  210  COEFE(K,J,LASTEDG) = 0.0
C
      IF(NMBRSEG(LASTEDG).EQ.1) GO TO 230
C
C---TRANSFER ETA MAXIMUMS
C
      DO 220 J=1,NMBRSEG(LASTEDG) - 1
C
        ETAMAX(J, NEWEDG) = ETAMAX(J,LASTEDG)
```

```
        ETAMAX(J,LASTEDG) = 0.0
C
C---TRANSFER SEGMENT MAXIMUMS
C
      DO 220 K=1,6
      SEGMAX(K,J, NEWEDG) = SEGMAX(K,J,LASTEDG)
  220 SEGMAX(K,J,LASTEDG) = 0.0
C
C---TRANSFER SURFACE MAPPING
C
  230     MAPSIDE(NEWSRF) = LMAPS(LASTSRF)
C
           IFLAGS(NEWSRF) = 1
C
C---TRANSFER SURFACE COEFFICIENTS
C
      DO 240 I=1,8
         COEFS(I, NEWSRF) = COEFS(I,LASTSRF)
  240    COEFS(I,LASTSRF) = 0.0
C
C---TRANSFER NUMBER OF NODES AND STRECHING DATA FOR ETA_2 AND ETA_3
C
      DO 250  I=2,IDIM
                       J = INDEX(I)
              NMBRNDS(J) = LNMBRND(J)
               STRTCH(J) = STRETCH(J)
  250          ISTRTCH(J) = LSTRTCH(J)
C-----------------------------------------------------------------------
C   READ INPUT PARAMETERS FOR STRETCHING FUNCTIONS
C-----------------------------------------------------------------------
  300 DO 310 I=1,IDIM
C
      STRETCH(I) = 0.0
C
      IF(ISTRTCH(I).LE.1) GO TO 310
C
      READ(NU5,1020) STRETCH
C
      GO TO 320
C
  310 CONTINUE
C-----------------------------------------------------------------------
C     COMPUTE STRETCHING FUNCTION PARAMETER B USING NEWTON-RAPHSON
C-----------------------------------------------------------------------
  320 DO 370 I=1,IDIM
C
          B = STRETCH(I)
         DS = B
      TNODE = REAL(NMBRNDS(I))
C
C---DECREASING OR INCREASING SPACING (INPUT MINIMUM SPACING)
C
      IF(ISTRTCH(I).EQ.5 .OR. ISTRTCH(I).EQ.6) THEN
C
                          B = SQRT(1.0 - (TNODE - 1.0)*DS)/(TNODE - 1.0)
C
                          DO 330 ITER=1,10
C
                            ARG1 = B*(TNODE - 1.)
                            ARG2 = B*(TNODE - 2.)
```

```
C
                        EXPA1 = EXP(ARG1)
                        EXPA2 = EXP(ARG2)
C
                        TANH1 = (EXPA1 - 1./EXPA1)/(EXPA1 + 1./EXPA1)
                        TANH2 = (EXPA2 - 1./EXPA2)/(EXPA2 + 1./EXPA2)
C
                          PSI = DS - (1.0 - TANH2/TANH1)
C
                       TANHP1 = 1.0 - TANH1**2
                       TANHP2 = 1.0 - TANH2**2
C
                         PSIP =   (1.0/TANH1)*(TANHP2*(TNODE - 2.)
     &                       - (TANH2/TANH1)* TANHP1*(TNODE - 1.))
C
                       IF(PSIP.EQ.0.) THEN
C
                                      WRITE(NU6,5100)
C
                                      STOP
C
                                      END IF
C
                         B0 = B
                          B = B0 - PSI/PSIP
                        DBF = (B - B0)/B0
C
C
  330                   IF(ABS(DBF).LE.0.001) GO TO 340
C
  340                                         END IF
C
C---DOUBLE STRETCHING (INPUT MINIMUM SPACING)
C
     IF(ISTRTCH(I).EQ.7) THEN
C
                       B = SQRT(1.0 - (TNODE - 1.0)*DS)/(TNODE - 1.0)
C
                       DO 350 ITER=1,10
C
                         ARG1 = B*(TNODE - 1.)
                         ARG3 = B*(TNODE - 3.)
C
                        EXPA1 = EXP(ARG1)
                        EXPA3 = EXP(ARG3)
C
                        TANH1 = (EXPA1 - 1./EXPA1)/(EXPA1 + 1./EXPA1)
                        TANH3 = (EXPA3 - 1./EXPA3)/(EXPA3 + 1./EXPA3)
C
                          PSI = DS - 0.5*(1.0 - TANH3/TANH1)
C
                       TANHP1 = 1.0 - TANH1**2
                       TANHP3 = 1.0 - TANH3**2
C
                         PSIP =   (0.5/TANH1)*(TANHP3*(TNODE - 3.0)
     &                       - (TANH3/TANH1)* TANHP1*(TNODE - 1.0))
C
                       IF(PSIP.EQ.0.) THEN
C
                                      WRITE(NU6,5000)
```

```
C
                                        STOP
C
                                        END IF
C
                          B0 = B
                           B = B0 - PSI/PSIP
                         DBF = (B - B0)/B0
C
C
  350                    IF(ABS(DBF).LE.0.001) GO TO 360
C
  360                    END IF
C
  370 STRETCH(I) = B
C
C---TRANSFER STRETCHING IN COMMON WITH PREVIOUS SECTION
C
      IF(ISECT.GT.1) THEN
C
                     DO 380 I=2,IDIM
  380                STRETCH(INDEX(I)) = STRTCH(INDEX(I))
C
                     END IF
C
C-----------------------------------------------------------------------
C   READ INPUT PARAMETERS FOR ARBITRARY GRID SPACING (ETAS)
C-----------------------------------------------------------------------
  400 DO 410 I=1,IDIM
C
      IF(ISTRTCH(I).NE.1)              GO TO 410
      IF(ISECT.GT.1 .AND. I.NE.MARCH) GO TO 410
C
      READ(NU5,1020)    (ETAS(I,J),J=1,NMBRNDS(I))
C
  410 CONTINUE
C-----------------------------------------------------------------------
C   READ INPUT PARAMETERS FOR EDGE COEFFICIENTS
C-----------------------------------------------------------------------
      WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
      WRITE(NU6,2040)
C
              LINE = 6
                II = 0
C
C---TOTAL NUMBER OF EDGES
C
                 NEDGES = 8*IDIM - 12
C
      DO 540 I=1,NEDGES
C
      IF(IFLAGE(I).EQ.1) GO TO 540
C
            ITOTAL = 1
               MAP = MAPEDGE(I)
C
C---DETERMINE THE NUMBER OF SEGMENTS ON AN EDGE
C
      DO 500 J=1,10
C
```

```
        NMBRSEG(I) = J
               MAP = MAP/10
C
      IF(MAP.EQ.0) GO TO 510
C
  500       ITOTAL = ITOTAL*10
C
  510          MAP = MAPEDGE(I)
C
      DO 530 J=1,NMBRSEG(I)
C
      DO 520 K=1,8
  520 COEFE(K,J,I) = 0.0
C
C---DETERMINE THE MAPPING FOR EACH SEGMENT
C
       MAPSEG(J,I) = MAP/ITOTAL
               MAP = MAP - MAPSEG(J,I)*ITOTAL
C
C---READ IN THE EDGE COEFFICIENTS FOR EACH SEGMENT
C
      IF(MAPSEG(J,I).LE.1) GO TO 530
C
                  READ(NU5,1020) (COEFE(K,J,I),K=1,8)
C
      IF(I.NE.II) THEN
C
                  LINE = LINE + 2
C
                  IF(LINE.GE.60) THEN
C
                                 WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
                                 WRITE(NU6,2040)
C
                                 LINE = 8
C
                                  END IF
C
                  WRITE(NU6,2050) I,J,(COEFE(K,J,I),K=1,8)
C
                  ELSE
C
                  LINE = LINE + 2
C
                  IF(LINE.GE.60) THEN
C
                                 WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
                                 WRITE(NU6,2040)
C
                                 LINE = 8
C
                                  END IF
C
                  WRITE(NU6,2060) J,(COEFE(K,J,I),K=1,8)
C
                  END IF
C
            II = I
C
  530   ITOTAL = ITOTAL/10
```

```
C
  540 CONTINUE
C-----------------------------------------------------------------------
C   READ INPUT PARAMETERS FOR SURFACE COEFFFICIENTS
C-----------------------------------------------------------------------
C
                      LINE = LINE + 5
C
                    IF(LINE.GE.60) THEN
C
                                WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
C
                                LINE = 6
C
                                   END IF
C
                    WRITE(NU6,3060)
C
C
      DO 610 I=1,6
C
C
      IF(IFLAGS(I).EQ.1) GO TO 610
C
      DO 600 J=1,8
  600 COEFS(J,I) = 0.0
C
      IF(MAPSIDE(I).LE.2) GO TO 610
C
      READ(NU5,1020) (COEFS(J,I),J=1,8)
C
                    LINE = LINE + 2
C
                  IF(LINE.GE.60) THEN
C
                                WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
                                WRITE(NU6,3060)
C
                                LINE = 8
C
                                END IF
C
      WRITE(NU6,3065) I,(COEFS(J,I),J=1,8)
C
  610 CONTINUE
C-----------------------------------------------------------------------
C   READ INPUT DATA FOR CORNER POINTS AND SEGMENT END POINTS
C-----------------------------------------------------------------------
                    LINE = LINE + 5
C
                  IF(LINE.GE.60) THEN
C
                                WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
C
                                LINE = 6
C
                                END IF
C
       WRITE(NU6,3070)
C
```

```
      DO 750 I=1,NEDGES
C
      IF(I.GT.8 .OR. IFLAGP(I).EQ.1) GO TO 700
C
C---READ IN CORNER POINTS
C
      READ(NU5,1020) (POINT(J,I),J=1,3)
C
         LINE = LINE + 2
C
      IF(LINE.GE.60) THEN
C
                    WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
                    WRITE(NU6,3070)
C
                      LINE = 8
C
                      END IF
C
       WRITE(NU6,3080) I,(POINT(J,I),J=1,3)
C
  700 IF(IFLAGE(I).EQ.1 .OR. NMBRSEG(I).EQ.1) GO TO 750
C
C---READ IN SEGMENT MAXIMUMS
C
      DO 740 J=1,NMBRSEG(I) - 1
C
      READ(NU5,1020) (SEGMAX(K,J,I),K=1,3),ETAMX
C
         LINE = LINE + 2
C
      IF(LINE.GE.60) THEN
C
                    WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
                    WRITE(NU6,3070)
C
                      LINE = 8
C
                      END IF
C
C
                     WRITE(NU6,3090) I,J,(SEGMAX(K,J,I),K=1,3),ETAMX
C
                     DO 720 N=1,3
                     DO 720 L=1,4
  720                IF(IRAY(L,N).EQ.I) GO TO 730
  730                CONTINUE
C
C
C---CONVERT NODE NUMBER TO ETA VALUE
C
            ETAMAX(J,I) = (ETAMX - 1.0)/(NMBRNDS(N) - 1.0)
C
C
  740 CONTINUE
  750 CONTINUE
C-----------------------------------------------------------------------
      RETURN
C
C---
```

```
 1000 FORMAT(16I5)
 1010 FORMAT(6I10)
 1020 FORMAT(8E10.0)
 1030 FORMAT((4I10))
C
 1100 FORMAT(1H1,10X,20A4,13X,8H SECTION,I2,3H OF,I3,9H FOR ZONE,I3)
 1110 FORMAT(// 11H FIXED ETA_,I1,8H VALUES: // (10(3X,F10.7)) )
 2040 FORMAT(// 25H EDGE SHAPE COEFFICIENTS:
     1       // 42H EDGE  SEGMENT       COEFF_1       COEFF_2,
     2          42H       COEFF_3       COEFF_4       COEFF_5,
     3          42H       COEFF_6       COEFF_7       COEFF_8 )
 2050 FORMAT(/   2X,I2,6X,I1,3X,8(1X,F13.7))
 2060 FORMAT(/        10X,I1,3X,8(1X,F13.7))
 2080 FORMAT(/   7H POINT ,I2,1H:,12X,2(7X,F13.7),9X,F7.2)
 2090 FORMAT(/   7H  EDGE ,I2,1H:,7X,I3,2X,2(7X,F13.7),9X,F7.2,9X,F9.2)
C
 3000 FORMAT(// 32H EDGE SHAPE FUNCTION INDICATORS:
     1       // 40H    EDGE_1    EDGE_2    EDGE_3    EDGE_4,
     2          40H    EDGE_5    EDGE_6    EDGE_7    EDGE_8,
     3          40H    EDGE_9   EDGE_10   EDGE_11   EDGE_12  / 12I10 )
 3010 FORMAT(// 35H SURFACE SHAPE FUNCTION INDICATORS:
     1       // 45H      SURFACE_1      SURFACE_2      SURFACE_3,
     2          45H      SURFACE_4      SURFACE_5      SURFACE_6
     3        /          6(10X,I5))
 3020 FORMAT(/// 26H MARCHING DIRECTION = ETA_,I1)
 3040 FORMAT(// 17H NUMBER OF NODES:
     1       // 30H     ETA_1     ETA_2     ETA_3 / 3I10
     2       // 29H NODAL STRETCHING INDICATORS:
     3       // 30H     ETA_1     ETA_2     ETA_3 / 3I10 )
 3050 FORMAT(// 32H STRETCHING FUNCTION PARAMETERS:
     1       // 45H           ETA_1           ETA_2           ETA_3
     2        /          3(2X,F13.7) )
 3060 FORMAT(// 28H SURFACE SHAPE COEFFICIENTS:
     1       // 42H SURFACE       COEFF_1       COEFF_2,
     2          42H       COEFF_3       COEFF_4       COEFF_5,
     3          42H       COEFF_6       COEFF_7       COEFF_8 )
 3065 FORMAT( /  4X,I1,9X,8(1X,F13.7))
 3070 FORMAT(//' COORDINATES :',
     1       //10X,' SEGMENT          X                Y',
     2                '              Z             ETAMAX')
 3080 FORMAT(/   7H POINT ,I2,1H:,8X,3(6X,F13.7))
 3090 FORMAT(/   7H  EDGE ,I2,1H:,3X,I3,2X,3(6X,F13.7)
     &                                                ,7X,F9.2)
 5000 FORMAT(1H0,'DEAD IN INPUT')
 5100 FORMAT(1H0,'DEAD IN INPUT')
C
      END
C
C*************************************************************************
C*****MAPPING*************************************************************
C*************************************************************************
C
      SUBROUTINE GRID
C-------------------------------------------------------------------------
C   GRID CONTROLS THE GENERATION OF THE SPATIAL COORDINATES OF EACH NODE
C   IN THE GRID.
C-------------------------------------------------------------------------
C               DOUBLE PRECISION NODE
                         REAL NODE
C
```

```
      COMMON /COUNTER/ NODESAV,NODETOT,NBNODES,NPLANE
      COMMON /HEADER/  ITITLE(20),LINE
      COMMON /INITA/   IDIM,MAPTEN,INCHES
      COMMON /INITB/   IBOX(4,6),IRAY(4,3),NRAY,IPT1(12),IPT2(12)
      COMMON /INPUTA/  EDGE(6,12),POINT(6,8),SIDE(6,6)
      COMMON /INPUTBC/ ISIDE(3)
      COMMON /IWRITE/  IWRTC,IWRTI,IWRTN
      COMMON /MAPING/  MAPSIDE(6),MAPSEG(10,12)
      COMMON /MARCHS/  MARCH,INDEX(3)
      COMMON /NODNMBR/ NODENUM(10000),TOL(3)
      COMMON /OUT/     NODE(5,10000)
      COMMON /PARTIAL/ DEDN(3,12),DSDN(3,2),SNORMAL(3,6)
      COMMON /SPACING/ ISTRTCH(3),STRETCH(3),ETAS(3,200),ETA(3),DETA(3)
      COMMON /UNITS/   NU5,NU6,NU19,NU20,NU21
      COMMON /ZONING/  ISECT,NSECT,IZONE,NMBRNDS(3)
C
      DIMENSION E(26)
      DIMENSION IFACE1(3,3),IFACE2(3,3),ISIDES(6)
      DIMENSION TANGENT(3),VECTOR(3),DIRECT(3)
C
      DATA IFACE1 /0,1,2,1,0,5,2,5,0/
      DATA IFACE2 /0,3,4,3,0,6,4,6,0/
      DATA ISIDES /3,4,1,2,6,5/
C
        INIT = 1
C
C
      DO 30 I=1,IDIM
C
      IF(STRETCH(I).LT.1.0) STRETCH(I) = 1.0
C
      DETA(I) = 1.0/(NMBRNDS(I) - 1.0)
C
                            FACTOR = NMBRNDS(I)*10.0
      IF(ISTRTCH(I).NE.0) FACTOR = NMBRNDS(I)*20.0
C
      DO 20 J=1,NRAY
C
           L = IRAY(J,I)
C
C---COMPARE X, Y, AND Z BETWEEN END POINTS OF AN EDGE
C
      DO 20 K=1,IDIM
      DELTA = ABS(POINT(K,IPT1(L)) - POINT(K,IPT2(L)))/FACTOR
C
      IF(DELTA.LE.0.0)    GO TO 20
      IF(DELTA.LT.TOL(K)) TOL(K) = DELTA
C
   20 CONTINUE
   30 CONTINUE
C
      TOL(1) = .005
      TOL(2) = .005
      TOL(3) = .005
C
C---DETERMINE SURFACES
C
      ISIDE1 = IFACE1(MARCH,INDEX(2))
      ISIDE2 = IFACE2(MARCH,INDEX(2))
      ISIDE3 = IFACE1(MARCH,INDEX(3))
```

```
        )
       ISIDE4 = IFACE2(MARCH,INDEX(3))
       ISIDE5 = IFACE1(INDEX(2),INDEX(3))
       ISIDE6 = IFACE2(INDEX(2),INDEX(3))
C
C---MAXIMUM NUMBER OF NODES IN A PLANE
C
          MAXPL = NMBRNDS(INDEX(2))*NMBRNDS(INDEX(3))
C
       INDEX(1) = MARCH
C-----------------------------------------------------------------------
C   AXIS
C-----------------------------------------------------------------------
       DO 700 IAXIS=1,NMBRNDS(INDEX(1))
C
       IF(IAXIS.EQ.1 .AND. ISECT.GT.1) GO TO 700
C
C---SEPERATE BOUNDARY CONDITIONS AND DETERMINE ETA
C
       CALL ETABC(MARCH,INDEX(1),IAXIS)
C
C---CALCULATE COORDINATES AND DERIVATIVES FOR POINTS ON EDGES
C
       CALL EDGES(INIT,INDEX(1),ETA(INDEX(1)))
C
C---NUMBER OF NODES TO BE STORED
C
       NODSTOR = NODESAV
C
C-----------------------------------------------------------------------
C   ROW
C-----------------------------------------------------------------------
       DO 500 JAXIS=1,NMBRNDS(INDEX(2))
C
C---SEPERATE BOUNDARY CONDITIONS AND DETERMINE ETA
C
       CALL ETABC(MARCH,INDEX(2),JAXIS)
C
C---CALCULATE COORDINATES AND DERIVATIVES FOR POINTS ON EDGES
C
       CALL EDGES(INIT,INDEX(2),ETA(INDEX(2)))
C
C
C---CALCULATE COORDINATES AND SURFACE NORMAL FOR POINTS ON SURFACES
C
       CALL SURFACE(INIT,ISIDE1)
C
       CALL SURFACE(INIT,ISIDE2)
C-----------------------------------------------------------------------
C   COLUMN
C-----------------------------------------------------------------------
       DO 400 KAXIS=1,NMBRNDS(INDEX(3))
C
C---SEPERATE BOUNDARY CONDITIONS AND DETERMINE ETA
C
       CALL ETABC(MARCH,INDEX(3),KAXIS)
C
C---CALCULATE COORDINATES AND DERIVATIVES FOR POINTS ON EDGES
C
       CALL EDGES(INIT,INDEX(3),ETA(INDEX(3)))
C
```

```
C---CALCULATE COORDINATES AND SURFACE NORMAL FOR POINTS ON SURFACES
C
      CALL SURFACE(INIT,ISIDE3)
C
      CALL SURFACE(INIT,ISIDE4)
C
      CALL SURFACE(INIT,ISIDE5)
C
      CALL SURFACE(INIT,ISIDE6)
C
C---ETA COEFFICIENTS FOR TRI-LINEAR INTERPOLATION
C
      E(1)  = 1.0 - ETA(3)
      E(2)  =       ETA(3)
      E(3)  = 1.0 - ETA(2)
      E(4)  =       ETA(2)
      E(5)  = 1.0 - ETA(1)
      E(6)  =       ETA(1)
      E(7)  =   E(5)*E(3)
      E(8)  =   E(5)*ETA(2)
      E(9)  = ETA(1)*E(3)
      E(10) = ETA(1)*ETA(2)
      E(11) =   E(5)*E(1)
      E(12) =   E(5)*ETA(3)
      E(13) = ETA(1)*E(1)
      E(14) = ETA(1)*ETA(3)
      E(15) =   E(3)*E(1)
      E(16) =   E(3)*ETA(3)
      E(17) = ETA(2)*E(1)
      E(18) = ETA(2)*ETA(3)
      E(19) =   E(5)*E(3)  *E(1)
      E(20) =   E(5)*E(3)  *ETA(3)
      E(21) =   E(5)*ETA(2)*E(1)
      E(22) =   E(5)*ETA(2)*ETA(3)
      E(23) = ETA(1)*E(3)  *E(1)
      E(24) = ETA(1)*E(3)  *ETA(3)
      E(25) = ETA(1)*ETA(2)*E(1)
      E(26) = ETA(1)*ETA(2)*ETA(3)
C
C---INCREMENT NODE COUNTERS
C
            INIT = 0
        NODESAV = NODESAV + 1
C
      NODNUM = NODESAV + NODETOT
C
C---CALCULATE THE COORDINATES
C
      DO 340 L=1,3
  340   NODE(L,NODESAV) = E(1)*SIDE(L,1) + E(2)*SIDE(L,3)
     1                  + E(3)*SIDE(L,2) + E(4)*SIDE(L,4)
     2                  + E(5)*SIDE(L,5) + E(6)*SIDE(L,6)
     3-E( 7)*EDGE(L,5)-E( 8)*EDGE(L, 8)-E( 9)*EDGE(L,6)-E(10)*EDGE(L, 7)
     4-E(11)*EDGE(L,4)-E(12)*EDGE(L,12)-E(13)*EDGE(L,2)-E(14)*EDGE(L,10)
     5-E(15)*EDGE(L,1)-E(16)*EDGE(L, 9)-E(17)*EDGE(L,3)-E(18)*EDGE(L,11)
     6                  + E(19)*POINT(L,1) + E(20)*POINT(L,5)
     7                  + E(21)*POINT(L,4) + E(22)*POINT(L,8)
     8                  + E(23)*POINT(L,2) + E(24)*POINT(L,6)
     9                  + E(25)*POINT(L,3) + E(26)*POINT(L,7)
C
```

```
C---INTERMEDIATE ERROR PRINT
C
      IF(IWRTI.EQ.0 .OR. MOD(NODNUM,IWRTI).NE.0) GO TO 350
C
                LINE = LINE + 27
C
                IF(LINE.GE.60) THEN
C
                                WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
C
                                LINE = 28
C
                                END IF
C
       WRITE(NU6,1110) NODNUM
      1                ,E(1),(SIDE(L,1),L=1,3)
      2                ,E(3),(SIDE(L,2),L=1,3)
      3                ,E(2),(SIDE(L,3),L=1,3)
      4                ,E(4),(SIDE(L,4),L=1,3)
      5                ,E(5),(SIDE(L,5),L=1,3)
      6                ,E(6),(SIDE(L,6),L=1,3)
C
       WRITE(NU6,1120) E(15),(EDGE(L,1),L=1,3)
      1                ,E(13),(EDGE(L,2),L=1,3)
      2                ,E(17),(EDGE(L,3),L=1,3)
      3                ,E(11),(EDGE(L,4),L=1,3)
      4                ,E( 7),(EDGE(L,5),L=1,3)
      5                ,E( 9),(EDGE(L,6),L=1,3)
C
       WRITE(NU6,1130) E(10),(EDGE(L,7),L=1,3)
      1                ,E( 8),(EDGE(L,8),L=1,3)
      2                ,E(16),(EDGE(L,9),L=1,3)
      3                ,E(14),(EDGE(L,10),L=1,3)
      4                ,E(18),(EDGE(L,11),L=1,3)
      5                ,E(12),(EDGE(L,12),L=1,3)
C
       WRITE(NU6,1140) E(19),(POINT(L,1),L=1,3)
      1                ,E(23),(POINT(L,2),L=1,3)
      2                ,E(25),(POINT(L,3),L=1,3)
      3                ,E(21),(POINT(L,4),L=1,3)
      4                ,E(20),(POINT(L,5),L=1,3)
      5                ,E(24),(POINT(L,6),L=1,3)
      6                ,E(26),(POINT(L,7),L=1,3)
      7                ,E(22),(POINT(L,8),L=1,3)
C
C
  350 CONTINUE
C
           NODE(4,NODESAV) = THETA
           NODE(5,NODESAV) = PHI
C
  400 CONTINUE
  500 CONTINUE
C-------------------------------------------------------------------
C   PLANE OUTPUT
C-------------------------------------------------------------------
      IF(IAXIS.EQ.1) GO TO 700
C
      IF(NODSTOR.NE.0) THEN
C
```

```
C---PRINT AND STORE DATA
C
      CALL OUTPUT(NU20,NODSTOR)
C
C---TRANSFER DATA SECOND PLANE TO FIRST PLANE
C
      DO 630 I=1,NODESAV
C
      DO 600 J=1,5
  600    NODE(J,I) = NODE(J,I + NODSTOR)
C
  630 CONTINUE
C
      END IF
C
C---TRANSFER NODE NUMBERS FROM SECOND PLANE TO FIRST PLANE
C
      DO 650 L=1,MAXPL
  650 NODENUM(L) = NODENUM(L + MAXPL)
C
C
  700 CONTINUE
C-----------------------------------------------------------------------
C   SECTION OUTPUT
C-----------------------------------------------------------------------
      IF(ISECT.NE.NSECT) RETURN
C
C---PRINT AND STORE DATA
                                        NODSTOR = NODESAV
C
      IF(NODESAV.NE.0) CALL OUTPUT(NU20,NODSTOR)
C
                                        NODESAV = 0
C
C
      RETURN
C
C---FORMAT STATEMENTS
C
 1100 FORMAT(1H1,10X,20A4,13X,8H SECTION,I2,3H OF,I3,9H FOR ZONE,I3)
C
 1110 FORMAT(  8H  NODE =,I10
     1       / 41X,2H X,13X,2H Y,12X,2H Z,20X,15H FLOW DIRECTION
     2       / 8H E( 1) =,F13.7,15H      SIDE(1) =,6(1X,F13.7)
     3       / 8H E( 3) =,F13.7,15H      SIDE(2) =,6(1X,F13.7)
     4       / 8H E( 2) =,F13.7,15H      SIDE(3) =,6(1X,F13.7)
     5       / 8H E( 4) =,F13.7,15H      SIDE(4) =,6(1X,F13.7)
     6       / 8H E( 5) =,F13.7,15H      SIDE(5) =,6(1X,F13.7)
     7       / 8H E( 6) =,F13.7,15H      SIDE(6) =,6(1X,F13.7))
C
 1120 FORMAT(  8H E(15) =,F13.7,15H     EDGE( 1) =,6(1X,F13.7)
     1       / 8H E(13) =,F13.7,15H     EDGE( 2) =,6(1X,F13.7)
     2       / 8H E(17) =,F13.7,15H     EDGE( 3) =,6(1X,F13.7)
     3       / 8H E(11) =,F13.7,15H     EDGE( 4) =,6(1X,F13.7)
     4       / 8H E( 7) =,F13.7,15H     EDGE( 5) =,6(1X,F13.7)
     5       / 8H E( 9) =,F13.7,15H     EDGE( 6) =,6(1X,F13.7))
C
 1130 FORMAT(  8H E(10) =,F13.7,15H     EDGE( 7) =,6(1X,F13.7)
     1       / 8H E( 8) =,F13.7,15H     EDGE( 8) =,6(1X,F13.7)
     2       / 8H E(16) =,F13.7,15H     EDGE( 9) =,6(1X,F13.7)
```

```
     3       / 8H E(14) =,F13.7,15H      EDGE(10) =,6(1X,F13.7)
     4       / 8H E(18) =,F13.7,15H      EDGE(11) =,6(1X,F13.7)
     5       / 8H E(12) =,F13.7,15H      EDGE(12) =,6(1X,F13.7))
C
 1140 FORMAT(  8H E(19) =,F13.7,15H      POINT(1) =,6(1X,F13.7)
     1       / 8H E(23) =,F13.7,15H      POINT(2) =,6(1X,F13.7)
     2       / 8H E(25) =,F13.7,15H      POINT(3) =,6(1X,F13.7)
     3       / 8H E(21) =,F13.7,15H      POINT(4) =,6(1X,F13.7)
     4       / 8H E(20) =,F13.7,15H      POINT(5) =,6(1X,F13.7)
     5       / 8H E(24) =,F13.7,15H      POINT(6) =,6(1X,F13.7)
     6       / 8H E(26) =,F13.7,15H      POINT(7) =,6(1X,F13.7)
     7       / 8H E(22) =,F13.7,15H      POINT(8) =,6(1X,F13.7))
C
      END
C
C*******************************************************************************
C*****MAPPING*******************************************************************
C*******************************************************************************
C
      SUBROUTINE EDGES(INIT,IDIR,ETA)
C-------------------------------------------------------------------------------
C   INTERPOLATES ALONG EACH EDGE OF EACH SIDE OF ZONE AS PART OF THE
C   BI/TRI-LINEAR INTERPOLATION SCHEME
C-------------------------------------------------------------------------------
C
      COMMON /COEFF/   COEFE(8,10,12),COEFS(8,6),NMBRSEG(12)
      COMMON /COUNTER/ NODESAV,NODETOT,NBNODES,NPLANE
      COMMON /HEADER/  ITITLE(20),LINE
      COMMON /INITA/   IDIM,MAPTEN,INCHES
      COMMON /INITB/   IBOX(4,6),IRAY(4,3),NRAY,IPT1(12),IPT2(12)
      COMMON /INPUTA/  EDGE(6,12),POINT(6,8),SIDE(6,6)
      COMMON /IWRITE/  IWRTC,IWRTI,IWRTN
      COMMON /MAPED/   KSEG(12),UAXIS(3,6),IEDGE1(6),IEDGE2(6),GAMMA
      COMMON /MAXIMUM/ ETAMAX(10,12),SEGMAX(6,10,12)
      COMMON /PARTIAL/ DEDN(3,12),DSDN(3,2),SNORMAL(3,6)
      COMMON /SEGMENT/ PT1(6,10,12),PT2(6,10,12),ETA1(10,12),ETA2(10,12)
      COMMON /UNITS/   NU5,NU6,NU19,NU20,NU21
      COMMON /ZONING/  ISECT,NSECT,IZONE,NMBRNDS(3)
C-------------------------------------------------------------------------------
C   INTERMEDIATE PRINT
C-------------------------------------------------------------------------------
      NODNUM = NODESAV + NODETOT + 1
C
      IF(IWRTI.EQ.0 .OR. MOD(NODNUM,IWRTI).NE.0) GO TO 20
C
                LINE = LINE + 2
C
                IF(LINE.GE.60) THEN
C
                               WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
C
                               LINE = 3
C
                               END IF
C
C-------------------------------------------------------------------------------
C   CALCULATE EDGE COORDINATES AND DERIVATIVE
C-------------------------------------------------------------------------------
   20 DO 200 I=1,NRAY
C
```

```
C---DETERMINE WHICH EDGE
C
           IEDGE = IRAY(I,IDIR)
C
C---EDGE INITIALIZATION
C
      IF(INIT.EQ.1) CALL EDGMAP1(IEDGE,IDIR,I)
C
C---DETERMINE WHICH SEGMENT
C
      DO 100 JSEG=1,NMBRSEG(IEDGE)
C
             ISEG = JSEG
C
  100 IF(ETA.GE.ETA1(JSEG,IEDGE) .AND. ETA.LE.ETA2(JSEG,IEDGE)) GOTO 110
C
  110 KSEG(IEDGE) = ISEG
C
C---DETERMINE WHERE ALONG THE SEGMENT
C
                       DENOM = ETA2(ISEG,IEDGE) - ETA1(ISEG,IEDGE)
      IF(DENOM.EQ.0.0) DENOM = 1.0
C
                       RATIO = (ETA - ETA1(ISEG,IEDGE))/DENOM
C
C---CALCULATE THE COORDINATES AND DERIVATIVE
C
      CALL EDGMAP2(IEDGE,ISEG,RATIO,EDGE(1,IEDGE),DEDN(1,IEDGE))
C-----------------------------------------------------------------------
C   INTERMEDIATE PRINT
C-----------------------------------------------------------------------
      IF(IWRTI.EQ.0 .OR. MOD(NODNUM,IWRTI).NE.0) GO TO 200
C
               LINE = LINE + 1
C
               IF(LINE.GE.60) THEN
C
                              WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
C
                              LINE = 2
C
                              END IF
C
C
               WRITE(NU6,1120) NODNUM,IEDGE,
     1                          (EDGE(J,IEDGE),J=1,3),THETA,PHI,
     2                          ETA,RATIO,ETAMAX(ISEG,IEDGE)
C
  200 CONTINUE
C
      RETURN
C
C---FORMAT STATEMENTS
C
 1100 FORMAT(1H1,10X,20A4,13X,8H SECTION,I2,3H OF,I3,9H FOR ZONE,I3)
 1120 FORMAT(1X,I6,4X,I2,3X,3(3X,F13.7),11X,2(2X,F7.2),3(2X,F7.5))
C
      END
C
C***********************************************************************
```

```
C*****MAPPING*****************************************************************
C******************************************************************************
C
      SUBROUTINE EDGMAP1(IEDGE,IDIR,NMBREDG)
C-------------------------------------------------------------------------
C   EDGE MAPPING INITIALIZATION
C-------------------------------------------------------------------------
C
      COMMON /COEFF/   COEFE(8,10,12),COEFS(8,6),NMBRSEG(12)
      COMMON /COUNTER/ NODESAV,NODETOT,NBNODES,NPLANE
      COMMON /EDGE0/   UI(3,10,12),UJ(3,10,12),UK(3,10,12),
     &                 R1(3,10,12),R2(3,10,12),THETA(10,12)
      COMMON /EDGE3/   ARC(10,12),ARC1(10,12),XLENGTH(10,12),
     &                 RA(10,12),RC(10,12),RE(10,12),THETA1(10,12)
      COMMON /EDGE8/   RM1(10,12),RM2(10,12),RPA1(10,12),RPA2(10,12)
      COMMON /HEADER/  ITITLE(20),LINE
      COMMON /INITA/   IDIM,MAPTEN,INCHES
      COMMON /INITB/   IBOX(4,6),IRAY(4,3),NRAY,IPT1(12),IPT2(12)
      COMMON /INITC/   PI,RADDEG
      COMMON /INPUTA/  EDGE(6,12),POINT(6,8),SIDE(6,6)
      COMMON /MAPED/   KSEG(12),UAXIS(3,6),IEDGE1(6),IEDGE2(6),GAMMA
      COMMON /MAPING/  MAPSIDE(6),MAPSEG(10,12)
      COMMON /MARCHS/  MARCH,INDEX(3)
      COMMON /MAXIMUM/ ETAMAX(10,12),SEGMAX(6,10,12)
      COMMON /PARTIAL/ DEDN(3,12),DSDN(3,2),SNORMAL(3,6)
      COMMON /SEGMENT/ PT1(6,10,12),PT2(6,10,12),ETA1(10,12),ETA2(10,12)
C
      DIMENSION ETAMX(10,12),VECTER(3)
C
C---INITIALIZE
C
      KSEG(IEDGE) = 1
      DO 10 N=1,NMBRSEG(IEDGE)
   10 ETAMX(N,IEDGE) = ETAMAX(N,IEDGE)
C
C
       ETAMX(NMBRSEG(IEDGE),IEDGE) = 1.0
C
                     ETA1(1,IEDGE) = 0.0
      ETAMAX(NMBRSEG(IEDGE),IEDGE) = 1.0
C-------------------------------------------------------------------------
C   INITIALIZE SEGMENTS
C-------------------------------------------------------------------------
      DO 1000 ISEG=1,NMBRSEG(IEDGE)
C
C---DETERMINE THE COORDINATES AT THE END POINTS
C
      IF(ISEG.EQ.1) THEN
C
      DO 50 J=1,6
                    PT1(J,   1,IEDGE) = POINT(J,IPT1(IEDGE))
      SEGMAX(J,NMBRSEG(IEDGE),IEDGE) = POINT(J,IPT2(IEDGE))
   50               PT2(J,ISEG,IEDGE) = SEGMAX(J,ISEG,IEDGE)
C
                     ELSE
C
      DO 60 J=1,6
                    PT1(J,ISEG,IEDGE) = SEGMAX(J,(ISEG-1),IEDGE)
   60               PT2(J,ISEG,IEDGE) = SEGMAX(J,ISEG,IEDGE)
C
```

```
                    END IF
C
C---ETA VALUE AT THE END OF EACH SEGMENT
C
      IF(ISEG.GT.1) ETA1(ISEG,IEDGE) = ETA2(ISEG-1,IEDGE)
                    ETA2(ISEG,IEDGE) = ETAMX(ISEG,IEDGE)
C
C---VECTORS FROM ORIGIN TO END POINTS
C
      DO 80 J=1,3
      R1(J,ISEG,IEDGE) = PT1(J,ISEG,IEDGE) - COEFE(J,ISEG,IEDGE)
   80 R2(J,ISEG,IEDGE) = PT2(J,ISEG,IEDGE) - COEFE(J,ISEG,IEDGE)
C
      CALL VMAG(R1(1,ISEG,IEDGE),RMAG1)
C
      CALL VMAG(R2(1,ISEG,IEDGE),RMAG2)
C-----------------------------------------------------------------------
C   CHOOSE MAPPING FUNCTION
C-----------------------------------------------------------------------
                          MAP = MAPSEG(ISEG,IEDGE)

      GO TO (1000,200,300,400,500) MAP
C-----------------------------------------------------------------------
C   CIRCULAR ARC                                                MAP = 2
C-----------------------------------------------------------------------
C---SWEEP ANGLE
C
  200 CALL VDOT(R1(1,ISEG,IEDGE),R2(1,ISEG,IEDGE),R1DOTR2)
C
                          ARG = R1DOTR2/(RMAG1*RMAG2)
C
      IF(ABS(ARG).GT.1.0) ARG = ARG/ABS(ARG)
C
            THETA(ISEG,IEDGE) = ACOS(ARG)
C
      GO TO 1000
C-----------------------------------------------------------------------
C   EDGE OF REVOLUTION                                          MAP = 4
C-----------------------------------------------------------------------
  300 CONTINUE
C
C---UK: NORMALIZED VECTOR ALONG AXIS OF REVOLUTION
C
      CALL VMAG(COEFE(4,ISEG,IEDGE),UKM)
C
      DO 410 I=1,3
C
  410 UK(I,ISEG,IEDGE) = COEFE(I + 3,ISEG,IEDGE)/UKM
C
C---UJ: NORMALIZED VECTOR FROM AXIS TOWARD EDGE
C
      CALL CROSS(UK(1,ISEG,IEDGE),R1(1,ISEG,IEDGE),UJ(1,ISEG,IEDGE),11)
C
C---UI: NORMALIZED VECTOR FROM AXIS TO FIRST POINT
C
      CALL CROSS(UJ(1,ISEG,IEDGE),UK(1,ISEG,IEDGE),UI(1,ISEG,IEDGE),12)
C
C---R1: VECTOR FROM AXIS TO FIRST POINT--------------------------------
C
      CALL VDOT(R1(1,ISEG,IEDGE),UK(1,ISEG,IEDGE),RPA1(ISEG,IEDGE))
```

```
C
      CALL VADD(1.0,R1(1,ISEG,IEDGE),-RPA1(ISEG,IEDGE),UK(1,ISEG,IEDGE),
     &               R1(1,ISEG,IEDGE),VECTER)
C
      CALL VMAG(R1(1,ISEG,IEDGE),RM1(ISEG,IEDGE))
C
C---R2: VECTOR FROM AXIS TO SECOND POINT-------------------------------
C
      CALL VDOT(R2(1,ISEG,IEDGE),UK(1,ISEG,IEDGE),RPA2(ISEG,IEDGE))
C
      CALL VADD(1.0,R2(1,ISEG,IEDGE),-RPA2(ISEG,IEDGE),UK(1,ISEG,IEDGE),
     &               R2(1,ISEG,IEDGE),VECTER)
C
      CALL VMAG(R2(1,ISEG,IEDGE),RM2(ISEG,IEDGE))
C
C---SWEEP ANGLE-------------------------------------------------------
C
      CALL VDOT(R1(1,ISEG,IEDGE),R2(1,ISEG,IEDGE),R1DOTR2)
C
                        ARG = R1DOTR2/(RM1(ISEG,IEDGE)*RM2(ISEG,IEDGE))
C
      IF(ABS(ARG).GT.1.) ARG = ARG/ABS(ARG)
C
           THETA(ISEG,IEDGE) = ACOS(ARG)
C
      GO TO 1000
C
  400 CONTINUE
C
C-----------------------------------------------------------------------
C   USER SUPPLIED SPECIAL EDGE INITIALIZATION                    MAP = 8
C-----------------------------------------------------------------------
C
      GO TO 1000
C
C-----------------------------------------------------------------------
C   USER SUPPLIED SPECIAL EDGE INITIALIZATION                    MAP = 9
C-----------------------------------------------------------------------
  500 CONTINUE
C
 1000 CONTINUE
C
      RETURN
      END
C
C***********************************************************************
C*****MAPPING***********************************************************
C***********************************************************************
C
      SUBROUTINE EDGMAP2(IEDGE,ISEG,RATIO,POINT,TANGENT)
C-----------------------------------------------------------------------
C   THIS ROUTINE CALCULATES THE COORDINATES
C   OF A POINT ON AN EDGE.
C-----------------------------------------------------------------------
C
      COMMON /COEFF/   COEFE(8,10,12),COEFS(8,6),NMBRSEG(12)
      COMMON /EDGE0/   UI(3,10,12),UJ(3,10,12),UK(3,10,12),
     &                 R1(3,10,12),R2(3,10,12),THETA(10,12)
      COMMON /EDGE3/   ARC(10,12),ARC1(10,12),XLENGTH(10,12),
     &                 RA(10,12),RC(10,12),RE(10,12),THETA1(10,12)
```

```
      COMMON /EDGE8/   RM1(10,12),RM2(10,12),RPA1(10,12),RPA2(10,12)
      COMMON /MAPING/  MAPSIDE(6),MAPSEG(10,12)
      COMMON /SEGMENT/ PT1(6,10,12),PT2(6,10,12),ETA1(10,12),ETA2(10,12)
      COMMON /ZONING/  ISECT,NSECT,IZONE,NMBRNDS(3)
C
      DIMENSION POINT(6),TANGENT(3)
      DIMENSION FLOWI(3),FLOWJ(3),FLOWK(3)
      DIMENSION VECTOR(3),C(3)
C
               MAP = MAPSEG(ISEG,IEDGE)
C
      GO TO (100,200,300,400,500) MAP
C-------------------------------------------------------------------------
C   LINEAR EDGE                                                    MAP = 1
C-------------------------------------------------------------------------
  100 DO 110 I=1,3
       POINT(I) = (1. - RATIO)*R1(I,ISEG,IEDGE) + RATIO*R2(I,ISEG,IEDGE)
C
  110 VECTOR(I) = R2(I,ISEG,IEDGE) - R1(I,ISEG,IEDGE)
C
      CALL VMAG(VECTOR,RMAG)
C
      IF(RMAG.EQ.0.0) RMAG = 1.0
C
      DO 120 I=1,3
  120 TANGENT(I) = (R2(I,ISEG,IEDGE) - R1(I,ISEG,IEDGE))/RMAG
C
      GO TO 1000
C-------------------------------------------------------------------------
C   CIRCULAR ARC                                                   MAP = 2
C-------------------------------------------------------------------------
  200       PHI1 = (1.0 - RATIO)*THETA(ISEG,IEDGE)
            PHI2 =         RATIO*THETA(ISEG,IEDGE)
C
      DO 210 I=1,3
        POINT(I) = COEFE(I,ISEG,IEDGE) +
     1          (SIN(PHI1)*R1(I,ISEG,IEDGE) + SIN(PHI2)*R2(I,ISEG,IEDGE))
     2          /SIN(THETA(ISEG,IEDGE))
C
  210 TANGENT(I) = THETA(ISEG,IEDGE)*
     1          (COS(PHI2)*R2(I,ISEG,IEDGE) - COS(PHI1)*R1(I,ISEG,IEDGE))
     2          /SIN(THETA(ISEG,IEDGE))
C
      GO TO 1000
C-------------------------------------------------------------------------
C   EDGE OF REVOLUTION                                             MAP = 4
C-------------------------------------------------------------------------
C---PROJECTION ALONG AXIS AND RADIUS
C
  300 RP = (RPA2(ISEG,IEDGE)-RPA1(ISEG,IEDGE))*RATIO + RPA1(ISEG,IEDGE)
      RM = ( RM2(ISEG,IEDGE)- RM1(ISEG,IEDGE))*RATIO +  RM1(ISEG,IEDGE)
C
C---ANGLE
C
      GAMMA = THETA(ISEG,IEDGE)*RATIO
C
C---CALCULATE THE POSITION AND TANGENT--------------------------------
C
      DO 410 I=1,3
C
```

```
              UR =  COS(GAMMA)*UI(I,ISEG,IEDGE)
     &             + SIN(GAMMA)*UJ(I,ISEG,IEDGE)
C
        POINT(I) =  COEFE(I,ISEG,IEDGE) + RP*UK(I,ISEG,IEDGE) + RM*UR
C
  410 TANGENT(I) =  COS(GAMMA)*UJ(I,ISEG,IEDGE)
     &             - SIN(GAMMA)*UI(I,ISEG,IEDGE)
C
      GO TO 1000
C
C-------------------------------------------------------------------------
C     HGM HOLE                                                    MAP = 8
C-------------------------------------------------------------------------
  400     IDUCT = IZONE - 3
C
      IF(IDUCT.EQ.1) THEN
C
C---UPPER DUCT
C
                IF(IEDGE.EQ.5) THEN
C
                               ICLOCK = 0
                                ANGLE = (1.0 - RATIO)*130.0 + RATIO*225.0
C
                               END IF
C
                IF(IEDGE.EQ.6) THEN
C
                               ICLOCK = 1
                                ANGLE = (1.0 - RATIO)*50.0 - RATIO*50.0
C
                               END IF
C
                IF(IEDGE.EQ.1) THEN
C
                               ICLOCK = 1
                                ANGLE = (1.0 - RATIO)*130.0 + RATIO*50.0
C
                               END IF
C
                IF(IEDGE.EQ.9) THEN
C
                               ICLOCK = 0
                                ANGLE = (1.0 - RATIO)*225.0 + RATIO*310.0
C
                               END IF
C
                     ELSE
C
C---LOWER DUCT
C
                IF(IEDGE.EQ.1) THEN
C
                               ICLOCK = 1
                                ANGLE = (1.0 - RATIO)*130.0 + RATIO*50.0
C
                               END IF
C
                IF(IEDGE.EQ.5) THEN
C
```

```
                              ICLOCK = 0
                               ANGLE = (1.0 - RATIO)*130.0 + RATIO*180.0
C
                              END IF
C
               IF(IEDGE.EQ.6) THEN
C
                              ICLOCK = 1
                               ANGLE = (1.0 - RATIO)*50.0
C
                              END IF
C
                        END IF
C
      CALL HOLES(ANGLE,POINT,TANGENT,ICLOCK,IDUCT)
C
      GO TO 1000
C-----------------------------------------------------------------------
C     HGM FAIRING                                                  MAP = 9
C-----------------------------------------------------------------------
  500 CALL FAIRING(PT1(1,ISEG,IEDGE),PT2(1,ISEG,IEDGE),POINT,TANGENT,C)
C
      GO TO 1000
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
 1000 CONTINUE
C
      RETURN
      END
C
C*************************************************************************
C*****MAPPING*************************************************************
C*************************************************************************
C
      SUBROUTINE SURFACE(INIT,ISIDE)
C-----------------------------------------------------------------------
C   THIS ROUTINE DETERMINES THE COORDINATES, FLOW VECTOR AND NORMAL OF A
C   POINT ON A SURFACE.
C-----------------------------------------------------------------------
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      COMMON /COEFF/   COEFE(8,10,12),COEFS(8,6),NMBRSEG(12)
      COMMON /COORD/   UA(3,6),UE(3,6),UN(3,6)
      COMMON /COUNTER/ NODESAV,MATESAV,NODETOT,NBNODES,NPLANE
      COMMON /HEADER/  ITITLE(20),LINE
      COMMON /HGM/     IHOLE
      COMMON /HOTGAS/  IHGM,ETA11,ETA12,ETA13,ETA31,ETA32,ETA33,
     &                 STR11,STR12,STR13,STR31,STR32,STR33,STR34
      COMMON /INITA/   IDIM,IFORMAT,MATRIX,MATING,MAPTEN,INCHES
      COMMON /INITB/   IBOX(4,6),IRAY(4,3),NRAY,IPT1(12),IPT2(12)
      COMMON /INITC/   PI,RADDEG
      COMMON /INPUTA/  EDGE(6,12),POINT(6,8),SIDE(6,6),INPUTBC(6)
      COMMON /IWRITE/  IWRTC,IWRTI,IWRTN
      COMMON /MARCHS/  MARCH,INDEX(3)
      COMMON /MAPED/   KSEG(12),UAXIS(3,6),IEDGE1(6),IEDGE2(6),GAMMA
      COMMON /MAPING/  MAPSIDE(6),MAPSEG(10,12)
      COMMON /MAXIMUM/ ETAMAX(10,12),SEGMAX(6,10,12)
      COMMON /PARTIAL/ DEDN(3,12),DSDN(3,2),SNORMAL(3,6)
      COMMON /SPACING/ ISTRTCH(3),STRETCH(3),ETAS(3,200),ETA(3),DETA(3)
      COMMON /UNITS/   NU5,NU6,NU19,NU20
```

```
      COMMON /ZONING/  ISECT,NSECT,IZONE,NMBRNDS(3)
C
      DIMENSION EDG(7,4),EDGX(4),EDGA(4),PT(7,4),PTX(4),PTA(4)
      DIMENSION ET1(4),ET3(4),SWEEP(3),VDUM(3)
      DIMENSION F(8),E1(3),E2(3),R1(3),R2(3)
      DIMENSION ZERO(3),VECTER(3)
      DIMENSION IETA1(6),IETA2(6),SIGNS(6)
      DIMENSION UNITRAD(3)
C
      DATA IETA1 /1,1,1,1,2,2/
      DATA IETA2 /2,3,2,3,3,3/
      DATA SIGNS /1.0,-1.0,-1.0,1.0,1.0,-1.0/
      DATA ZERO  /0.,0.,0./
C
      NODNUM = NODESAV + NODETOT + 1
C
C---ETA COEFFICIENTS FOR BI-LINEAR INTERPOLATION
C
      F(1) = 1.0 - ETA(IETA1(ISIDE))
      F(2) =       ETA(IETA1(ISIDE))
      F(3) = 1.0 - ETA(IETA2(ISIDE))
      F(4) =       ETA(IETA2(ISIDE))
      F(5) = F(1)*F(3)
      F(6) = F(1)*F(4)
      F(7) = F(2)*F(3)
      F(8) = F(2)*F(4)
C
C---DETERMINE THE EDGES OF THE SURFACE
C
      LINE1 = IBOX(1,ISIDE)
      LINE2 = IBOX(2,ISIDE)
      LINE3 = IBOX(3,ISIDE)
      LINE4 = IBOX(4,ISIDE)
C
C---DETERMINE THE CORNER POINTS OF THE SURFACE
C
      LPT1 = IPT1(LINE1)
      LPT2 = IPT1(LINE3)
      LPT3 = IPT2(LINE3)
      LPT4 = IPT2(LINE1)
C
                                              MAP = MAPSIDE(ISIDE)
C
      GO TO (100,100,300,400,500,600,700,800) MAP
C-------------------------------------------------------------------
C   FLAT OR CYLINDRICAL SURFACE                          MAP = 1 OR 2
C-------------------------------------------------------------------
C---CALCULATE POSITION
C
  100 DO 110 J=1,3
C
          SIDE(J,ISIDE) =  F(1)*EDGE(J,LINE1) - F(5)*POINT(J,LPT1)
     1                   + F(2)*EDGE(J,LINE3) - F(6)*POINT(J,LPT4)
     2                   + F(3)*EDGE(J,LINE2) - F(7)*POINT(J,LPT2)
     3                   + F(4)*EDGE(J,LINE4) - F(8)*POINT(J,LPT3)
C
              DSDN(J,1) =       EDGE(J,LINE3) + F(3)*POINT(J,LPT1)
     1                        - EDGE(J,LINE1) + F(4)*POINT(J,LPT4)
     2                   + F(3)*DEDN(J,LINE2) - F(3)*POINT(J,LPT2)
     3                   + F(4)*DEDN(J,LINE4) - F(4)*POINT(J,LPT3)
```

```
C
  110        DSDN(J,2) =       EDGE(J,LINE4) + F(1)*POINT(J,LPT1)
     1                      - EDGE(J,LINE2) + F(2)*POINT(J,LPT2)
     2                  + F(1)*DEDN(J,LINE1) - F(1)*POINT(J,LPT4)
     3                  + F(2)*DEDN(J,LINE3) - F(2)*POINT(J,LPT3)
C
      CALL CROSS(DSDN(1,1),DSDN(1,2),SNORMAL(1,ISIDE),40)
C
      GO TO 1000
C-------------------------------------------------------------------------
C   SPECIAL SURFACE                                                MAP = 3
C-------------------------------------------------------------------------
  300 GO TO 1000
C-------------------------------------------------------------------------
C   EDGE OF REVOLUTION                                             MAP = 4
C-------------------------------------------------------------------------
  400 IF(INIT.EQ.1) THEN
C
C---ETA DIRECTION OF REVOLUTION
C
                                          IONETWO = 1
      IF(COEFS(7,ISIDE).EQ.IETA2(ISIDE)) IONETWO = 2
C
C---EDGE 1
C
            IEDGE1(ISIDE) = IBOX(IONETWO,ISIDE)
C
C---EDGE 2
C
            IEDGE2(ISIDE) = IBOX(IONETWO + 2,ISIDE)
C
C---UA: NORMALIZED VECTOR ALONG AXIS OF REVOLUTION
C
      CALL VADD(1.0,COEFS(4,ISIDE),1.0,ZERO,VECTER,UA(1,ISIDE))
C
C---UN: NORMALIZED VECTOR FROM AXIS TOWARD SURFACE
C
      CALL VADD(1.0,EDGE(1,IEDGE1(ISIDE)),-1.0,COEFS(1,ISIDE),E1,VECTER)
C
      CALL CROSS(UA(1,ISIDE),E1,UN(1,ISIDE),41)
C
C---UE: NORMALIZED VECTOR FROM AXIS TO FIRST EDGE
C
      CALL CROSS(UN(1,ISIDE),UA(1,ISIDE),UE(1,ISIDE),42)
C
                  END IF
C
C---R1: VECTOR FROM AXIS TO FIRST EDGE---------------------------------
C
      CALL VADD(1.0,EDGE(1,IEDGE1(ISIDE)),-1.0,COEFS(1,ISIDE),E1,VECTER)
C
      CALL VDOT(E1,UA(1,ISIDE),EPA1)
C
      CALL VADD(1.0,E1,-EPA1,UA(1,ISIDE),R1,VECTER)
C
      CALL VMAG(R1,RM1)
C
C---DERIVATIVE
C
      CALL VDOT(DEDN(1,IEDGE1(ISIDE)),UA(1,ISIDE),DA1)
```

```
C
      CALL VDOT(DEDN(1,IEDGE1(ISIDE)),VECTER,DR1)
C
      CALL VMAG(DEDN(1,IEDGE1(ISIDE)),DM1)
C
C---R2: VECTOR FROM AXIS TO SECOND EDGE--------------------------------
C
      CALL VADD(1.0,EDGE(1,IEDGE2(ISIDE)),-1.0,COEFS(1,ISIDE),E2,VECTER)
C
      CALL VDOT(E2,UA(1,ISIDE),EPA2)
C
      CALL VADD(1.0,E2,-EPA2,UA(1,ISIDE),R2,VECTER)
C
      CALL VMAG(R2,RM2)
C
C---DERIVATIVE
C
      CALL VDOT(DEDN(1,IEDGE2(ISIDE)),UA(1,ISIDE),DA2)
C
      CALL VDOT(DEDN(1,IEDGE2(ISIDE)),VECTER,DR2)
C
      CALL VMAG(DEDN(1,IEDGE2(ISIDE)),DM2)
C
C---SWEEP ANGLE-------------------------------------------------------
C
      CALL VDOT(R1,R2,R1DOTR2)
C
                          ARG = R1DOTR2/(RM1*RM2)
C
      IF(ABS(ARG).GT.1.0) ARG = ARG/ABS(ARG)
C
      THETA = ACOS(ARG)
C
C---CALCULATE RADIUS--------------------------------------------------
C
          N = COEFS(7,ISIDE)
C
C---POSITION
C
          EP = (EPA2 - EPA1)*ETA(N) + EPA1
          RM = (RM2  - RM1 )*ETA(N) + RM1
C
C---DERIVATIVE
C
          DA = (DA2 - DA1)*ETA(N) + DA1
          DR = (DR2 - DR1)*ETA(N) + DR1
          DM = (DM2 - DM1)*ETA(N) + DM1
C
C---ANGLE
C
      GAMMA = THETA*ETA(N)
C
C---CALCULATE THE POSITION AND SURFACE NORMAL-------------------------
C
      DO 410 I=1,3
C
                UR =  COS(GAMMA)*UE(I,ISIDE) + SIN(GAMMA)*UN(I,ISIDE)
C
      SIDE(I,ISIDE) =  COEFS(I,ISIDE) + EP*UA(I,ISIDE) + RM*UR
C
```

```
          DSDN(I,1) =  (DR*UR + DA*UA(I,ISIDE))/DM
C
  410     DSDN(I,2) =  COS(GAMMA)*UN(I,ISIDE) - SIN(GAMMA)*UE(I,ISIDE)
C
C---ORIENT SURFACE NORMAL DEPENDING ON DIRECTION OF EDGE REVOLUTION
C
      IF(COEFS(7,ISIDE).EQ.IETA2(ISIDE)) THEN
C
      CALL CROSS(DSDN(1,1),DSDN(1,2),SNORMAL(1,ISIDE),43)
C
                                         ELSE
C
      CALL CROSS(DSDN(1,2),DSDN(1,1),SNORMAL(1,ISIDE),44)
C
                                         END IF
C
      GO TO 1000
C-----------------------------------------------------------------------
C     BOWL SURFACE WITH HOLES                                    MAP = 5
C-----------------------------------------------------------------------
  500    ETA1 = F(2)
       ET1(1) = (ETA11 - 1.0) / REAL(NMBRNDS(1) - 1)
       ET1(2) = (ETA12 - 1.0) / REAL(NMBRNDS(1) - 1)
       ET1(3) = (ETA13 - 1.0) / REAL(NMBRNDS(1) - 1)
       ET1(4) =  1.0
C
                           IET1 = 1
      IF(ETA1.GE.ET1(1)) IET1 = 2
      IF(ETA1.GE.ET1(2)) IET1 = 3
      IF(ETA1.GE.ET1(3)) IET1 = 4
C
         ETA3 = F(4)
       ET3(1) = (ETA31 - 1.0) / REAL(NMBRNDS(3) - 1)
       ET3(2) = (ETA32 - 1.0) / REAL(NMBRNDS(3) - 1)
       ET3(3) = (ETA33 - 1.0) / REAL(NMBRNDS(3) - 1)
       ET3(4) =  1.0
C
                           IET3 = 1
      IF(ETA3.GE.ET3(1)) IET3 = 2
      IF(ETA3.GE.ET3(2)) IET3 = 3
      IF(ETA3.GE.ET3(3)) IET3 = 4
C
       IHOLE = 0
      ICLOCK = 0
C
      GO TO (510,520,540,560) IET1
C-----------------------------------------------------------------------
C     IET1 = 1
C-----------------------------------------------------------------------
  510 CALL CONVERT(EDGE(1, 3),EDGX(1),EDGA(1))
C
      CALL CONVERT(EDGE(1,11),EDGX(3),EDGA(3))
C
           RANGE = EDGX(1)
C
           ANGLE = (1.0 - ETA3)*EDGA(1) + ETA3*EDGA(3)
C
      GO TO 580
C-----------------------------------------------------------------------
C     IET1 = 2
```

```
C-----------------------------------------------------------------------
  520     RATIO1 = (ETA1 - ET1(1)) / (ET1(2) - ET1(1))
C
             X1 = RATIO1*STR11/2.0
          ETAX1 = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
             X2 = STR11/2.0
          ETAX2 = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
           EPS1 = ETAX1/ETAX2
C
      GO TO (522,524,526,528) IET3
C
C---IET3 = 1-----------------------------------------------------------
C
  522     RATIO3 = ETA3/ET3(1)
C
             X1 = RATIO3*STR31/2.0
          ETAX1 = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
             X2 = STR31/2.0
          ETAX2 = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
           EPS3 = ETAX1/ETAX2
C
      CALL CONVERT(EDGE(1,3),EDGX(1),EDGA(1))
C
      CALL CONVERT(EDGE(1,8),EDGX(4),EDGA(4))
C
                EDGX(4) = SEGMAX(1,1,3) - 6.07
C
                  IDUCT = 1
                PT(7,3) = 130.0
C
      CALL HOLES(PT(7,3),PT(1,3),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(PT(1,3),PTX(3),PTA(3))
C
        EDGX(3) = (1.0 - EPS1)*EDGX(4) + EPS1*PTX(3)
C
          RANGE = (1.0 - EPS3)*EDGX(1) + EPS3*EDGX(3)
C
        EDGA(2) = EPS3*PTA(3)
C
          ANGLE = (1.0 - EPS1)*EDGA(4) + EPS1*EDGA(2)
C
      GO TO 580
C
C---IET3 = 2-----------------------------------------------------------
C
  524  RATIO3 = (ETA3 - ET3(1))/(ET3(2) - ET3(1))
C
           X1 = (STR32/2.0)/2.0
       ETAMID = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
           X2 = (RATIO3 - 0.5)*STR32/2.0
        ETAMX = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
         EPS3 = (ETAMX + ETAMID)/(2.0*ETAMID)
C
```

```
        CALL CONVERT(EDGE(1,8),EDGX(4),EDGA(4))
C
                  EDGX(4) = SEGMAX(1,1,3) - 6.07
C
                    IDUCT = 1
                 EDG(7,2) = (1.0 - EPS3)*130.0 + EPS3*225.0
C
      CALL HOLES(EDG(7,2),EDG(1,2),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(EDG(1,2),EDGX(2),EDGA(2))
C
          RANGE = (1.0 - EPS1)*EDGX(4) + EPS1*EDGX(2)
C
          ANGLE = (1.0 - EPS1)*EDGA(4) + EPS1*EDGA(2)
C
      GO TO 580
C
C---IET3 = 3-------------------------------------------------------------
C
  526  RATIO3 = (ETA3 - ET3(2))/(ET3(3) - ET3(2))
C
           X1 = (STR33/2.0)/2.0
       ETAMID = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
           X2 = (RATIO3 - 0.5)*STR33/2.0
        ETAMX = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
         EPS3 = (ETAMX + ETAMID)/(2.0*ETAMID)
C
      CALL CONVERT(EDGE(1,8),EDGX(4),EDGA(4))
C
                  EDGX(4) = SEGMAX(1,1,3) - 6.07
C
                    IDUCT = 1
                 PT(7,2) = 225.0
C
      CALL HOLES(PT(7,2),PT(1,2),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(PT(1,2),PTX(2),PTA(2))
C
                    IDUCT = 2
                 PT(7,3) = 130.0
C
      CALL HOLES(PT(7,3),PT(1,3),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(PT(1,3),PTX(3),PTA(3))
C
        EDGX(2) = (1.0 - EPS3)*PTX(2) + EPS3*PTX(3)
C
          RANGE = (1.0 - EPS1)*EDGX(4) + EPS1*EDGX(2)
C
        EDGA(2) = (1.0 - EPS3)*PTA(2) + EPS3*PTA(3)
C
          ANGLE = (1.0 - EPS1)*EDGA(4) + EPS1*EDGA(2)
C
      GO TO 580
C
C---IET3 = 4-------------------------------------------------------------
C
  528  RATIO3 = 1.0 - (ETA3 - ET3(3))/(ET3(4) - ET3(3))
```

```
C
           X1 = RATIO3*STR34/2.0
         ETAX1 = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
           X2 = STR34/2.0
         ETAX2 = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
          EPS3 = 1.0 - ETAX1/ETAX2
C
      CALL CONVERT(EDGE(1,8),EDGX(4),EDGA(4))
C
                  EDGX(4) = SEGMAX(1,1,3) - 6.07
C
                    IDUCT = 2
                 EDG(7,2) = (1.0 - EPS3)*130.0 + EPS3*180.0
C
      CALL HOLES(EDG(7,2),EDG(1,2),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(EDG(1,2),EDGX(2),EDGA(2))
C
          RANGE = (1.0 - EPS1)*EDGX(4) + EPS1*EDGX(2)
C
          ANGLE = (1.0 - EPS1)*EDGA(4) + EPS1*EDGA(2)
C
      GO TO 580
C-----------------------------------------------------------------------
C     IET1 = 3
C-----------------------------------------------------------------------
  540  RATIO1 = (ETA1 - ET1(2))/(ET1(3) - ET1(2))
C
           X1 = (STR12/2.0)/2.0
        ETAMID = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
           X2 = (RATIO1 - 0.5)*STR12/2.0
         ETAMX = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
          EPS1 = (ETAMX + ETAMID)/(2.0*ETAMID)
C
      GO TO (542,544,546,548) IET3
C
C---IET3 = 1-----------------------------------------------------------
C
  542     RATIO3 = ETA3/ET3(1)
C
              X1 = RATIO3*STR31/2.0
            ETAX1 = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
              X2 = STR31/2.0
            ETAX2 = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
             EPS3 = ETAX1/ETAX2
C
      CALL CONVERT(EDGE(1,3),EDGX(1),EDGA(1))
C
                    IDUCT = 1
                 EDG(7,3) = (1.0 - EPS1)*130.0 + EPS1*50.0
C
      CALL HOLES(EDG(7,3),EDG(1,3),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(EDG(1,3),EDGX(3),EDGA(3))
```

C-2

```
C
          RANGE = (1.0 - EPS3)*EDGX(1) + EPS3*EDGX(3)
C
          ANGLE = (1.0 - EPS3)*EDGA(1) + EPS3*EDGA(3)
C
      GO TO 580
C
C---IET3 = 2-------------------------------------------------------------
C
  544   RATIO3 = (ETA3 - ET3(1))/(ET3(2) - ET3(1))
C
             X1 = (STR32/2.0)/2.0
         ETAMID = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
             X2 = (RATIO3 - 0.5)*STR32/2.0
          ETAMX = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
           EPS3 = (ETAMX + ETAMID)/(2.0*ETAMID)
C
          IHOLE = 1
          IDUCT = 1
C
       EDG(7,1) = (1.0 - EPS1)*130.0 + EPS1*50.0
       EDG(7,2) = (1.0 - EPS3)*50.0  + EPS3*(-50.0)
       EDG(7,3) = (1.0 - EPS1)*225.0 + EPS1*310.0
       EDG(7,4) = (1.0 - EPS3)*130.0 + EPS3*225.0
C
        PT(7,1) = 130.0
        PT(7,2) =  50.0
        PT(7,3) = 310.0
        PT(7,4) = 225.0
C
       DO 545 N=1,4
C
       CALL HOLES(EDG(7,N),EDG(1,N),VDUM,ICLOCK,IDUCT)
C
       CALL CONVERT(EDG(1,N),EDGX(N),EDGA(N))
C
       CALL HOLES(PT(7,N),PT(1,N),VDUM,ICLOCK,IDUCT)
C
  545 CALL CONVERT(PT(1,N),PTX(N),PTA(N))
C
       RANGE = (1.0 - EPS3)*EDGX(1) - (1.0 - EPS1)*(1.0 - EPS3)*PTX(1)
      1              + EPS1*EDGX(2)          - (1.0 - EPS3)*EPS1*PTX(2)
      2              + EPS3*EDGX(3)                   - EPS1*EPS3*PTX(3)
      3       + (1.0 - EPS1)*EDGX(4)          - (1.0 - EPS1)*EPS3*PTX(4)
C
       ANGLE = (1.0 - EPS3)*EDGA(1) - (1.0 - EPS1)*(1.0 - EPS3)*PTA(1)
      1              + EPS1*EDGA(2)          - (1.0 - EPS3)*EPS1*PTA(2)
      2              + EPS3*EDGA(3)                   - EPS1*EPS3*PTA(3)
      3       + (1.0 - EPS1)*EDGA(4)          - (1.0 - EPS1)*EPS3*PTA(4)
C
       GO TO 580
C
C---IET3 = 3-------------------------------------------------------------
C
  546  RATIO3 = (ETA3 - ET3(2))/(ET3(3) - ET3(2))
C
             X1 = (STR33/2.0)/2.0
         ETAMID = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
```

```
C
           X2 = (RATIO3 - 0.5)*STR33/2.0
         ETAMX = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
          EPS3 = (ETAMX + ETAMID)/(2.0*ETAMID)
C
                    IDUCT = 1
                 EDG(7,1) = (1.0 - EPS1)*225.0 + EPS1*310.0
C
      CALL HOLES(EDG(7,1),EDG(1,1),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(EDG(1,1),EDGX(1),EDGA(1))
C
                    IDUCT = 2
                 EDG(7,3) = (1.0 - EPS1)*130.0 + EPS1*50.0
C
      CALL HOLES(EDG(7,3),EDG(1,3),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(EDG(1,3),EDGX(3),EDGA(3))
C
          RANGE = (1.0 - EPS3)*EDGX(1) + EPS3*EDGX(3)
C
          ANGLE = (1.0 - EPS3)*EDGA(1) + EPS3*EDGA(3)
C
      GO TO 580
C
C---IET3 = 4------------------------------------------------------------
C
  548  RATIO3 = 1.0 - (ETA3 - ET3(3))/(ET3(4) - ET3(3))
C
           X1 = RATIO3*STR34/2.0
         ETAX1 = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
           X2 = STR34/2.0
         ETAX2 = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
          EPS3 = 1.0 - ETAX1/ETAX2
C
          IHOLE = 1
          IDUCT = 2
C
      EDG(7,1) = (1.0 - EPS1)*130.0 + EPS1*50.0
      EDG(7,2) = (1.0 - EPS3)*50.0
      EDG(7,4) = (1.0 - EPS3)*130.0 + EPS3*180.0
C
       PT(7,1) = 130.0
       PT(7,2) =  50.0
       PT(7,3) =   0.0
       PT(7,4) = 180.0
C
      DO 549 N=1,4
C
      CALL HOLES(EDG(7,N),EDG(1,N),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(EDG(1,N),EDGX(N),EDGA(N))
C
      CALL HOLES(PT(7,N),PT(1,N),VDUM,ICLOCK,IDUCT)
C
  549 CALL CONVERT(PT(1,N),PTX(N),PTA(N))
C
```

```
        EDGX(3) = (1.0 - EPS1)*PTX(4) + EPS1*PTX(3)
C
        EDGA(3) = (1.0 - EPS1)*PTA(4) + EPS1*PTA(3)
C
      RANGE = (1.0 - EPS3)*EDGX(1) - (1.0 - EPS1)*(1.0 - EPS3)*PTX(1)
     1               + EPS1*EDGX(2)          - (1.0 - EPS3)*EPS1*PTX(2)
     2               + EPS3*EDGX(3)                  - EPS1*EPS3*PTX(3)
     3       + (1.0 - EPS1)*EDGX(4)          - (1.0 - EPS1)*EPS3*PTX(4)
C
      ANGLE = (1.0 - EPS3)*EDGA(1) - (1.0 - EPS1)*(1.0 - EPS3)*PTA(1)
     1               + EPS1*EDGA(2)          - (1.0 - EPS3)*EPS1*PTA(2)
     2               + EPS3*EDGA(3)                  - EPS1*EPS3*PTA(3)
     3       + (1.0 - EPS1)*EDGA(4)          - (1.0 - EPS1)*EPS3*PTA(4)
C
      GO TO 580
C-----------------------------------------------------------------------
C     IET1 = 4
C-----------------------------------------------------------------------
  560  RATIO1 = (ETA1 - ET1(3))/(ET1(4) - ET1(3))
C
           X1 = (STR13/2.0)/2.0
       ETAMID = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
           X2 = (RATIO1 - 0.5)*STR13/2.0
        ETAMX = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
         EPS1 = (ETAMX + ETAMID)/(2.0*ETAMID)
C
      GO TO (562,564,566,568) IET3
C
C---IET3 = 1-----------------------------------------------------------
C
  562     RATIO3 = ETA3/ET3(1)
C
              X1 = RATIO3*STR31/2.0
           ETAX1 = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
              X2 = STR31/2.0
           ETAX2 = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
            EPS3 = ETAX1/ETAX2
C
      CALL CONVERT(EDGE(1,3),EDGX(1),EDGA(1))
C
      CALL CONVERT(EDGE(1,7),EDGX(2),EDGA(2))
C
                  IDUCT = 1
                PT(7,4) = 50.0
C
      CALL HOLES(PT(7,4),PT(1,4),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(PT(1,4),PTX(4),PTA(4))
C
        EDGX(3) = (1.0 - EPS1)*PTX(4)  + EPS1*EDGX(2)
C
          RANGE = (1.0 - EPS3)*EDGX(1) + EPS3*EDGX(3)
C
        EDGA(4) = EPS3*PTA(4)
C
          ANGLE = (1.0 - EPS1)*EDGA(4) + EPS1*EDGA(2)
```

```
C
      GO TO 580
C
C---IET3 = 2------------------------------------------------------------
C
  564  RATIO3 = (ETA3 - ET3(1))/(ET3(2) - ET3(1))
C
           X1 = (STR32/2.0)/2.0
        ETAMID = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
           X2 = (RATIO3 - 0.5)*STR32/2.0
         ETAMX = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
          EPS3 = (ETAMX + ETAMID)/(2.0*ETAMID)
C
      CALL CONVERT(EDGE(1,7),EDGX(2),EDGA(2))
C
                    IDUCT = 1
                EDG(7,4) = (1.0 - EPS3)*50.0 + EPS3*(-50.0)
C
      CALL HOLES(EDG(7,4),EDG(1,4),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(EDG(1,4),EDGX(4),EDGA(4))
C
          RANGE = (1.0 - EPS1)*EDGX(4) + EPS1*EDGX(2)
C
          ANGLE = (1.0 - EPS1)*EDGA(4) + EPS1*EDGA(2)
C
      GO TO 580
C
C---IET3 = 3------------------------------------------------------------
C
  566  RATIO3 = (ETA3 - ET3(2))/(ET3(3) - ET3(2))
C
           X1 = (STR33/2.0)/2.0
        ETAMID = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
           X2 = (RATIO3 - 0.5)*STR33/2.0
         ETAMX = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
          EPS3 = (ETAMX + ETAMID)/(2.0*ETAMID)
C
      CALL CONVERT(EDGE(1,7),EDGX(2),EDGA(2))
C
                    IDUCT = 1
                PT(7,1) = 310.0
C
      CALL HOLES(PT(7,1),PT(1,1),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(PT(1,1),PTX(1),PTA(1))
C
                    IDUCT = 2
                PT(7,4) = 50.0
C
      CALL HOLES(PT(7,4),PT(1,4),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(PT(1,4),PTX(4),PTA(4))
C
        EDGX(4) = (1.0 - EPS3)*PTX(1) + EPS3*PTX(4)
C
```

```
          RANGE = (1.0 - EPS1)*EDGX(4) + EPS1*EDGX(2)
C
        EDGA(4) = (1.0 - EPS3)*PTA(1) + EPS3*PTA(4)
C
          ANGLE = (1.0 - EPS1)*EDGA(4) + EPS1*EDGA(2)
C
      GO TO 580
C
C---IET3 = 4------------------------------------------------------------
C
  568  RATIO3 = 1.0 - (ETA3 - ET3(3))/(ET3(4) - ET3(3))
C
             X1 = RATIO3*STR34/2.0
          ETAX1 = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
             X2 = STR34/2.0
          ETAX2 = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
           EPS3 = 1.0 - ETAX1/ETAX2
C
      CALL CONVERT(EDGE(1,7),EDGX(2),EDGA(2))
C
                     IDUCT = 2
                  EDG(7,4) = (1.0 - EPS3)*50.0
C
      CALL HOLES(EDG(7,4),EDG(1,4),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(EDG(1,4),EDGX(4),EDGA(4))
C
          RANGE = (1.0 - EPS1)*EDGX(4) + EPS1*EDGX(2)
C
          ANGLE = (1.0 - EPS1)*EDGA(4) + EPS1*EDGA(2)
C
      GO TO 580
C
  580 CONTINUE
C
C---RADIUS AND TANGENT-------------------------------------------------
C
                                       XX = RANGE + 6.07
C
C---FIRST SEGMENT
C
      IF(XX.LE.SEGMAX(1,1,3)) THEN
C
                                            RAD = 7.434
C
                                  DSDN(1,1) = 1.0
                                  DSDN(2,1) = 0.0
                                  DSDN(3,1) = 0.0
C
                                  GO TO 590
C
                                  END IF
C
C---SECOND SEGMENT
C
      IF(XX.LT.SEGMAX(1,2,3)) THEN
C
                                  RAD = SQRT(7.890**2 - (3.375 - RANGE)**2)
```

```
C
                                        TH = ACOS(RAD/7.890)
                                IF(XX.GT.9.445) TH = -TH
C
                                DSDN(1,1) = COS(TH)
                                DSDN(2,1) = SIN(TH)
                                DSDN(3,1) = 0.0
C
                                GO TO 590
C
                                END IF
C
C---LAST SEGMENT
C
                                      RAD = 14.5636605 - 0.57735027*XX
C
                                       TH = -30.0*RADDEG
C
                                DSDN(1,1) =  COS(TH)
                                DSDN(2,1) =  SIN(TH)
                                DSDN(3,1) =  0.0
C
C---POSITION-----------------------------------------------------------------
C
  590 SWEEP(1) = 0.0
      SWEEP(2) = COS(ANGLE)
      SWEEP(3) = SIN(ANGLE)
C
      UA(1,ISIDE) = 1.0
      UA(2,ISIDE) = 0.0
      UA(3,ISIDE) = 0.0
C
      CALL VADD(XX,UA(1,ISIDE),RAD,SWEEP,SIDE(1,ISIDE),VDUM)
C
C---NORMAL-------------------------------------------------------------------
C
C---TANGENT 1
C
             DA = DSDN(1,1)
             DN = DSDN(2,1)
C
      CALL VADD(DN,SWEEP,DA,UA(1,ISIDE),VDUM,DSDN(1,1))
C
C---TANGENT 2
C
      DSDN(1,2) =  0.0
      DSDN(2,2) = -SIN(ANGLE)
      DSDN(3,2) =  COS(ANGLE)
C
C---SURFACE NORMAL
C
      CALL CROSS(DSDN(1,1),DSDN(1,2),SNORMAL(1,ISIDE),45)
C
      GO TO 1030
C-----------------------------------------------------------------------------
C     MATED SURFACE OF DUCT                                          MAP = 6
C-----------------------------------------------------------------------------
  600     EPS1 = F(2)
          EPS3 = F(4)
C
```

```
         IDUCT = IZONE - 3
C
      IF(IDUCT.EQ.1) THEN
C
C---UPPER DUCT-------------------------------------------------------------
C
      EDG(7,1) = (1.0 - EPS1)*130.0 + EPS1*50.0
      EDG(7,2) = (1.0 - EPS3)*50.0  + EPS3*(-50.0)
      EDG(7,3) = (1.0 - EPS1)*225.0 + EPS1*310.0
      EDG(7,4) = (1.0 - EPS3)*130.0 + EPS3*225.0
C
       PT(7,1) = 130.0
       PT(7,2) =  50.0
       PT(7,3) = 310.0
       PT(7,4) = 225.0
C
                      ELSE
C
C---LOWER DUCT-------------------------------------------------------------
C
      EDG(7,1) = (1.0 - EPS1)*130.0 + EPS1*50.0
      EDG(7,2) = (1.0 - EPS3)*50.0
      EDG(7,4) = (1.0 - EPS3)*130.0 + EPS3*180.0
C
      PT(7,1) = 130.0
      PT(7,2) =  50.0
      PT(7,3) =   0.0
      PT(7,4) = 180.0
C
                      END IF
C
C---CALCULATE POINTS AND EDGES---------------------------------------------
C
      DO 610 N=1,4
C
      CALL HOLES(EDG(7,N),EDG(1,N),VDUM,ICLOCK,IDUCT)
C
      CALL CONVERT(EDG(1,N),EDGX(N),EDGA(N))
C
      CALL HOLES(PT(7,N),PT(1,N),VDUM,ICLOCK,IDUCT)
C
  610 CALL CONVERT(PT(1,N),PTX(N),PTA(N))
C
      IF(IDUCT.EQ.2) THEN
C
                      EDGX(3) = (1.0 - EPS1)*PTX(4) + EPS1*PTX(3)
C
                      EDGA(3) = (1.0 - EPS1)*PTA(4) + EPS1*PTA(3)
C
                      END IF
C
C---DISTANCE ALONG AXIS AND ANGLE FROM VERTICAL
C
      RANGE = (1.0 - EPS3)*EDGX(1) - (1.0 - EPS1)*(1.0 - EPS3)*PTX(1)
     1              + EPS1*EDGX(2)          - (1.0 - EPS3)*EPS1*PTX(2)
     2              + EPS3*EDGX(3)                   - EPS1*EPS3*PTX(3)
     3       + (1.0 - EPS1)*EDGX(4)          - (1.0 - EPS1)*EPS3*PTX(4)
C
      ANGLE = (1.0 - EPS3)*EDGA(1) - (1.0 - EPS1)*(1.0 - EPS3)*PTA(1)
     1              + EPS1*EDGA(2)          - (1.0 - EPS3)*EPS1*PTA(2)
```

```
     2              + EPS3*EDGA(3)                    - EPS1*EPS3*PTA(3)
     3      + (1.0 - EPS1)*EDGA(4)          - (1.0 - EPS1)*EPS3*PTA(4)
C
C---POSITION------------------------------------------------------------
C
                   RAD = SQRT(7.890**2 - (3.375 - RANGE)**2)
C
         SIDE(1,ISIDE) = RANGE + 6.07
         SIDE(2,ISIDE) = RAD*COS(ANGLE)
         SIDE(3,ISIDE) = RAD*SIN(ANGLE)
C
C---NORMAL--------------------------------------------------------------
C
               VDUM(1) = SIDE(1,ISIDE) - 9.445
               VDUM(2) = SIDE(2,ISIDE)
               VDUM(3) = SIDE(3,ISIDE)
C
      CALL VMAG(VDUM,SMAG)
C
      SNORMAL(1,ISIDE) = VDUM(1)/SMAG
      SNORMAL(2,ISIDE) = VDUM(2)/SMAG
      SNORMAL(3,ISIDE) = VDUM(3)/SMAG
C
      GO TO 1030
C-----------------------------------------------------------------------
C     OUTER SURFACE OF DUCT                                      MAP = 7
C-----------------------------------------------------------------------
  700 IF(ISIDE.LT.4) THEN
C
C---SURFACE 1 AND 3
                     M = LINE2
                     N = LINE4
C
                     ELSE
C---SURFACE 5 AND 6
                     M = LINE1
                     N = LINE3
C
                     END IF
C
      CALL FAIRING(EDGE(1,M),EDGE(1,N),SIDE(1,ISIDE),
     *             VDUM,SNORMAL(1,ISIDE))
C
      GO TO 1030
C-----------------------------------------------------------------------
C   SPECIAL SURFACE                                              MAP = 8
C-----------------------------------------------------------------------
C
  800 GO TO 1000
C-----------------------------------------------------------------------
C   DIRECT SURFACE NORMAL INTO FLOW DOMAIN
C-----------------------------------------------------------------------
 1000 DO 1010 I=1,3
 1010 SNORMAL(I,ISIDE) = SNORMAL(I,ISIDE)*SIGNS(ISIDE)
C
C-----------------------------------------------------------------------
C   ERROR MESSAGE IF SURFACE NORMAL = 0
C-----------------------------------------------------------------------
 1030 CALL VMAG(SNORMAL(1,ISIDE),SNMAG)
C
```

```
      IF(SNMAG.EQ.0.0) THEN
C
                       LINE = LINE + 5
C
                       IF (LINE.GE.60) THEN
C
                                WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
C
                                LINE = 6
C
                                       END IF
C
                       WRITE(NU6,1110)
C
C
C
                       RETURN
C
                       END IF
C-----------------------------------------------------------------------
C   INTERMEDIATE PRINT
C-----------------------------------------------------------------------
      IF(IWRTI.EQ.0 .OR. MOD(NODNUM,IWRTI).NE.0) RETURN
C
                LINE = LINE + 3
C
                IF(LINE.GE.60) THEN
C
                                WRITE(NU6,1100) ITITLE,ISECT,NSECT,IZONE
C
                                LINE = 4
C
                                END IF
C
C
C
      RETURN
C
C---FORMAT STATEMENTS
C
 1100 FORMAT(1H1,10X,20A4,13X,8H SECTION,I2,3H OF,I3,9H FOR ZONE,I3)
 1110 FORMAT(/ 49H SURFACE NORMAL EQUALS ZERO IN SUBROUTINE SURFACE)
C
      END
C
C**************************************************************************
C*****HGM*****************************************************************
C**************************************************************************
C
      SUBROUTINE CONVERT(POINT,RANGE,ANGLE)
C-----------------------------------------------------------------------
C     CALCULATES PROJECTION AND SWEEP ANGLE OF POINT ON BOWL SURFACE
C
C       POINT = POSITION ON BOWL SURFACE
C       RANGE = PROJECTION ONTO X AXIS FROM EDGE OF BOWL
C       ANGLE = SWEEP ANGLE FROM VERTICAL
C-----------------------------------------------------------------------
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION POINT(3),BOWL(3),AXIS(3),VECTOR(3),SWEEP(3),VDUM(3)
```

```
C
      DATA BOWL /6.07,0.0,0.0/
      DATA AXIS /1.0,0.0,0.0/
C
C---DISTANCE ALONG AXIS FROM EDGE OF BOWL
C
           ONE = 1.0
C
      CALL VADD(ONE,POINT,-ONE,BOWL,VECTOR,VDUM)
C
      CALL VDOT(VECTOR,AXIS,RANGE)
C
C---ANGLE FROM VERTICAL
C
      CALL VADD(ONE,VECTOR,-RANGE,AXIS,VDUM,SWEEP)
C
         ANGLE = ACOS(SWEEP(2))
C
      RETURN
      END
C
C*************************************************************************
C*****OUTPUT**************************************************************
C*************************************************************************
C
      SUBROUTINE BLKOUT(NUNIT,NODSTOR)
C-------------------------------------------------------------------
C   WRITES THE FORMATTED BLOCKED GEOMETRY FILE (NUNIT)
C-------------------------------------------------------------------
C               DOUBLE PRECISION NODE
                          REAL NODE
C
      COMMON /INITA/    IDIM,MAPTEN,INCHES
      COMMON /IOCOUNT/ IREWIND(40),NREAD(40),NWRITE(40)
      COMMON /OUT/      NODE(5,10000)
      COMMON /ZONING/   ISECT,NSECT,IZONE,NMBRNDS(3)
      COMMON /MARCHS/   MARCH,INDEX(3)
C
      IPLANE = NWRITE(NUNIT) + 1
C
           WRITE(NUNIT,1000)  NODSTOR,IPLANE,(NMBRNDS(I),I=1,3),MARCH
           WRITE(NUNIT,1010) (NODE(1,I),I=1,NODSTOR)
           WRITE(NUNIT,1010) (NODE(2,I),I=1,NODSTOR)
           WRITE(NUNIT,1010) (NODE(3,I),I=1,NODSTOR)
C
      RETURN
C
C---FORMAT STATEMENTS
C
 1000 FORMAT(24I5)
 1010 FORMAT(10E13.7)
C
      END
C
C
C*************************************************************************
C*****REWRITE*************************************************************
C*************************************************************************
C
      SUBROUTINE REWRITE
```

```
C-----------------------------------------------------------------
C
C     THIS PROGRAM MAKES THE HGM GEOMETRY OUTPUT FILE
C     COMPATABLE WITH PLOT3D GEOMETRY FILES
C
C-----------------------------------------------------------------
C
      COMMON /ZONING/  ISECT,NSECT,IZONE,NMBRNDS(3)
      COMMON /UNITS/   NU5,NU6,NU19,NU20,NU21
      COMMON /MARCHS/  MARCH,INDEX(3)
C
      DIMENSION ID(10),JD(10),KD(10)
      DIMENSION XBUF(155000),YBUF(155000),ZBUF(155000)
C
      REWIND (NU20)
C
      I2 = 0
      NGRID = 0
      READ(NU20,9000,END=200)NSTORE,IPLN,INOD2,JNOD2,KNOD2,
     *                       MARCH2
C
 9000 FORMAT(8I5)
C
      WRITE(NU6,9000)NSTORE,IPLN,INOD2,JNOD2,KNOD2,MARCH2
C
      I1 = I2 + 1
      I2 = I2 + NSTORE
C
      READ(NU20,9010,ERR=400)(XBUF(I),I=I1,I2)
 9010 FORMAT(10E13.7)
      READ(NU20,9010,ERR=400)(YBUF(I),I=I1,I2)
      READ(NU20,9010,ERR=400)(ZBUF(I),I=I1,I2)
C
C
  100 READ(NU20,9000,END=200)NSTORE,IPLN,INOD,JNOD,KNOD,MARCH
C
      WRITE(NU6,9000)NSTORE,IPLN,INOD,JNOD,KNOD,MARCH
C
C
      IF(INOD.NE.INOD2.OR.JNOD.NE.JNOD2.OR.KNOD.NE.KNOD2)THEN
      NGRID = NGRID + 1
C
      IF(MARCH2.EQ.1)THEN
      ID(NGRID) = KNOD2
      JD(NGRID) = JNOD2
      KD(NGRID) = INOD2
      END IF
C
      IF(MARCH2.EQ.2)THEN
      ID(NGRID) = INOD2
      JD(NGRID) = KNOD2
      KD(NGRID) = JNOD2
      END IF
C
      IF(MARCH2.EQ.3)THEN
      ID(NGRID) = JNOD2
      JD(NGRID) = INOD2
      KD(NGRID) = KNOD2
      END IF
C
```

```
      WRITE(NU6,9120)NGRID,ID(NGRID),JD(NGRID),KD(NGRID),I2
 9120 FORMAT(' NGRID,IDIM,JDIM,KDIM,I2 ',5I5)
C
      WRITE(19,9010,ERR=450)
     1                (XBUF(I),I=1,I2),
     2                (YBUF(I),I=1,I2),
     3                (ZBUF(I),I=1,I2)
C
      INOD2 = INOD
      JNOD2 = JNOD
      KNOD2 = KNOD
      MARCH2 = MARCH
      I2 = 0
C
C
      END IF
C
C
      I1 = I2 + 1
      I2 = I2 + NSTORE
C
      READ(NU20,9010,ERR=400)(XBUF(I),I=I1,I2)
      READ(NU20,9010,ERR=400)(YBUF(I),I=I1,I2)
      READ(NU20,9010,ERR=400)(ZBUF(I),I=I1,I2)
C
      GOTO 100
C
C
C
  200 CONTINUE
C
C
C
      WRITE(NU6,9115)
 9115 FORMAT(' END OF FILE REACHED ON FILE 20')
C
      NGRID = NGRID + 1
C
      IF(MARCH2.EQ.1)THEN
      ID(NGRID) = KNOD2
      JD(NGRID) = JNOD2
      KD(NGRID) = INOD2
      END IF
C
      IF(MARCH2.EQ.2)THEN
      ID(NGRID) = INOD2
      JD(NGRID) = KNOD2
      KD(NGRID) = JNOD2
      END IF
C
      IF(MARCH2.EQ.3)THEN
      ID(NGRID) = JNOD2
      JD(NGRID) = INOD2
      KD(NGRID) = KNOD2
      END IF
C
C      ID(NGRID) = INOD2
C      JD(NGRID) = JNOD2
C      KD(NGRID) = KNOD2
C
```

```
      WRITE(NU6,9120)NGRID,ID(NGRID),JD(NGRID),KD(NGRID),I2
      WRITE(NU6,8000)INOD2,JNOD2,KNOD2
 8000 FORMAT(' INOD2,JNOD2,KNOD2 ',3I5)
C
      WRITE(19,9010,ERR=450)
     1                (XBUF(I),I=1,I2),
     2                (YBUF(I),I=1,I2),
     3                (ZBUF(I),I=1,I2)
C
      WRITE(NU6,8100)
 8100 FORMAT(' GRID WRITTEN TO 19 FILE')
C
      REWIND(19)
C
      IF(NGRID.GT.1)THEN
      WRITE(NU6,8300)NGRID
 8300 FORMAT(' NGRID ',I5)
      WRITE(NU21,ERR=475)NGRID
      END IF
C
      WRITE(NU6,8350)(ID(N),JD(N),KD(N),N=1,NGRID)
 8350 FORMAT(' IDIM,JDIM,KDIM ',3I5)
C
      WRITE(NU21,ERR=475)(ID(N),JD(N),KD(N),N=1,NGRID)
C
C
      DO 300 N=1,NGRID
      I2 = ID(N)*JD(N)*KD(N)
C
      READ(19,9010,ERR=450)
     1                (XBUF(I),I=1,I2),
     2                (YBUF(I),I=1,I2),
     3                (ZBUF(I),I=1,I2)
C
      WRITE(NU21,ERR=475)
     1                (XBUF(I),I=1,I2),
     2                (YBUF(I),I=1,I2),
     3                (ZBUF(I),I=1,I2)
C
      WRITE(NU6,9130)N
 9130 FORMAT(' GRID ',I5,' WRITTEN TO XYZHGM.DAT')
C
  300 CONTINUE
C
C
      WRITE(NU6,9150)NGRID
 9150 FORMAT(' THERE ARE ',I5,' GRIDS WRITTEN TO XYZHGM.DAT')
      GOTO 500
C
  400 WRITE(NU6,9140)
 9140 FORMAT('ERROR TRYING TO READ THE 20 FILE')
      GOTO 500
C
  450 WRITE(NU6,9160)
 9160 FORMAT('ERROR TRYING TO READ FILE 19')
      GOTO 500
C
  475 WRITE(NU6,9170)
 9170 FORMAT(' ERROR TRYING TO WRITE TO 20 FILE')
C
```

```
  500 CONTINUE
C
      RETURN
C
      END
C
C
C*************************************************************************
C*****HGM*****************************************************************
C*************************************************************************
C
      SUBROUTINE PROJECT(POINT,RANGE,ANGLE)
C-------------------------------------------------------------------------
C     CALCULATES PROJECTION AND SWEEP ANGLE OF POINT ON BOWL SURFACE
C
C       POINT = POSITION ON BOWL SURFACE
C       RANGE = PROJECTION ONTO X AXIS FROM EDGE OF BOWL
C       ANGLE = SWEEP ANGLE FROM VERTICAL
C-------------------------------------------------------------------------
C
      DIMENSION POINT(3),BOWL(3),AXIS(3),VECTOR(3),SWEEP(3),VDUM(3)
C
      DATA BOWL /6.07,0.0,0.0/
      DATA AXIS /1.0,0.0,0.0/
C
C---DISTANCE ALONG AXIS FROM EDGE OF BOWL
C
           ONE = 1.0
C
      CALL VADD(ONE,POINT,-ONE,BOWL,VECTOR,VDUM)
C
      CALL VDOT(VECTOR,AXIS,RANGE)
C
C---ANGLE FROM VERTICAL
C
      CALL VADD(ONE,VECTOR,-RANGE,AXIS,VDUM,SWEEP)
C
         ANGLE = ACOS(SWEEP(2))
C
      RETURN
      END
C
C*************************************************************************
C*****UTILITY*************************************************************
C*************************************************************************
C
      SUBROUTINE CROSS(A,B,C,N)
C-------------------------------------------------------------------------
C  C = CROSS PRODUCT OF A AND B (UNIT VECTOR)
C-------------------------------------------------------------------------
C
      COMMON /COUNTER/ NODESAV,NODETOT,NBNODES,NPLANE
      COMMON /HEADER/  ITITLE(20),LINE
      COMMON /INITA/   IDIM,MAPTEN,INCHES
      COMMON /IWRITE/  IWRTC,IWRTI,IWRTN
      COMMON /UNITS/   NU5,NU6,NU19,NU20,NU21
      COMMON /ZONING/  ISECT,NSECT,IZONE,NMBRNDS(3)
C
      DIMENSION A(3),B(3),C(3)
C
```

```
C---CROSS PRODUCT
C
      C(1) = A(2)*B(3) - A(3)*B(2)
      C(2) = A(3)*B(1) - A(1)*B(3)
      C(3) = A(1)*B(2) - A(2)*B(1)
C
C---MAGNITUDE
C
      CALL VMAG(C,CMAG)
C
      IF(CMAG.GT.0.0) THEN
C
C---NORMALIZE
C
      C(1) = C(1)/CMAG
      C(2) = C(2)/CMAG
      C(3) = C(3)/CMAG
C
                        ELSE
      C(1) = 0.0
      C(2) = 0.0
      C(3) = 0.0
C
      IF(IWRTI.EQ.0) RETURN
C-------------------------------------------------------------------------
C   ERROR PRINT (CROSS PRODUCT EQUALS ZERO)
C
C   N = 10 -> 21  SUBROUTINE EDGMAP1
C       30 -> 32       "       EDGMAP2
C       40 -> 45       "       SURFACE
C       50 -> 56       "       BC
C       60 -> 63       "       HOLES
C       70 -> 73       "       FAIRING
C-------------------------------------------------------------------------
                                  NODNUM = NODESAV + NODETOT
         IF(N.LT.50 .OR. N.GT.59) NODNUM = NODNUM + 1
C
                        LINE = LINE + 1
C
                        IF(LINE.GE.60) THEN
C
                                  WRITE(NU6,1000) ITITLE,ISECT,NSECT,I2ONE
C
                                  LINE = 2
C
                                       END IF
C
                        WRITE(NU6,1010) N,NODNUM
C
                        END IF
      RETURN
C
C---FORMAT STATEMENTS
C
 1000 FORMAT(1H1,10X,20A4,13X,8H SECTION,I2,3H OF,I3,9H FOR ZONE,I3)
 1010 FORMAT(9H LOCATION,I3,36H: CROSS PRODUCT EQUALS ZERO FOR NODE,I6)
C
      END
C
C*************************************************************************
```

```
C*****GRID SPACING*****************************************************************
C***********************************************************************************
C
      SUBROUTINE ETABC(MARCH,INDEX,NODE)
C---------------------------------------------------------------------
C
C   THIS ROUTINE CACULATES THE VALUE OF ETA.
C
C---------------------------------------------------------------------
C
      COMMON /INITA/    IDIM,MAPTEN,INCHES
      COMMON /INITC/    PI,RADDEG
      COMMON /INPUTA/   EDGE(6,12),POINT(6,8),SIDE(6,6)
      COMMON /INPUTBC/ ISIDE(3)
      COMMON /SPACING/ ISTRTCH(3),STRETCH(3),ETAS(3,200),ETA(3),DETA(3)
      COMMON /ZONING/   ISECT,NSECT,IZONE,NMBRNDS(3)
C---------------------------------------------------------------------
C   FIRST NODE
C---------------------------------------------------------------------
      IF(NODE.EQ.1) THEN
C
                    ETA(INDEX) = 0.0
C
C---DETERMINE SIDE
C
                    ISIDE(INDEX) = 10 + (INDEX - 6)*INDEX
C
C---STORE SPACING
C
                    ETAS(INDEX,1) = 0.0
C
                    RETURN
C
                    END IF
C---------------------------------------------------------------------
C   LAST NODE
C---------------------------------------------------------------------
      IF(NODE.EQ.NMBRNDS(INDEX)) THEN
C
                                  ETA(INDEX) = 1.0
C
C---DETERMINE SIDE
C
                    ISIDE(INDEX) = 9 + (INDEX - 7)*INDEX/2
C
C---STORE SPACING
C
                                  ETAS(INDEX,NODE) = 1.0
C
                                  RETURN
C
                                  END IF
C---------------------------------------------------------------------
C   INTERIOR NODES
C---------------------------------------------------------------------
C
     IF(ISECT.GT.1 .AND. INDEX.NE.MARCH) GO TO 310
C
C---CALCULATE ETA
C
```

```
                ISTR = ISTRTCH(INDEX) + 1
C
      GO TO (100,110,120,130,140,150,160,170,180,190,200) ISTR
C
C---EQUAL SPACING-------------------------------------------------------(0)
C
  100 ETA(INDEX) = ETA(INDEX) + DETA(INDEX)
C
      GO TO 300
C
C---INPUT ETA SPACING---------------------------------------------------(1)
C
  110 ETA(INDEX) = ETAS(INDEX,NODE)
C
      GO TO 300
C
C---DECREASING SPACING; INPUT STRETCHING FACTOR-----------------------(2)
C
  120       RATIO = REAL(NODE - 1)/REAL(NMBRNDS(INDEX) - 1)
C
              X1 = RATIO*STRETCH(INDEX)/2.0
            ETA1 = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
              X2 = STRETCH(INDEX)/2.0
            ETA2 = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
      ETA(INDEX) = ETA1/ETA2
C
      GO TO 300
C
C---INCREASING SPACING; INPUT STRETCHING FACTOR-----------------------(3)
C
  130       RATIO = REAL(NMBRNDS(INDEX) - NODE)/REAL(NMBRNDS(INDEX) - 1)
C
              X1 = RATIO*STRETCH(INDEX)/2.0
            ETA1 = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
              X2 = STRETCH(INDEX)/2.0
            ETA2 = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
      ETA(INDEX) = 1.0 - ETA1/ETA2
C
      GO TO 300
C
C---DOUBLE STRETCHING; INPUT STRETCHING FACTOR------------------------(4)
C
  140         X1 = (STRETCH(INDEX)/2.0)/2.0
           ETAMID = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
            RATIO = REAL(NODE - 1)/REAL(NMBRNDS(INDEX) - 1)
C
              X2 = (RATIO - 0.5)*STRETCH(INDEX)/2.0
            ETAMX = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
      ETA(INDEX) = (ETAMX + ETAMID)/(2.0*ETAMID)
C
      GO TO 300
C
C---DECREASING SPACING; INPUT MINIMUM SPACING-------------------------(5)
C
```

```
  150       ARGI = STRETCH(INDEX)*REAL(NODE - 1)
            EXPI = EXP(ARGI)
           EXPII = 1.0/EXPI
           TANHI = (EXPI - EXPII)/(EXPI + EXPII)
C
            ARGN = STRETCH(INDEX)*REAL(NMBRNDS(INDEX) - 1)
            EXPN = EXP(ARGN)
           EXPNI = 1.0/EXPN
           TANHN = (EXPN - EXPNI)/(EXPN + EXPNI)
C
      ETA(INDEX) = TANHI/TANHN
C
      GO TO 300
C
C---INCREASING SPACING; INPUT MINIMUM SPACING-----------------------(6)
C
  160       ARGI = STRETCH(INDEX)*REAL(NMBRNDS(INDEX) - NODE)
            EXPI = EXP(ARGI)
           EXPII = 1.0/EXPI
           TANHI = (EXPI - EXPII)/(EXPI + EXPII)
C
            ARGN = STRETCH(INDEX)*REAL(NMBRNDS(INDEX) - 1)
            EXPN = EXP(ARGN)
           EXPNI = 1.0/EXPN
           TANHN = (EXPN - EXPNI)/(EXPN + EXPNI)
C
      ETA(INDEX) = 1.0 - TANHI/TANHN
C
      GO TO 300
C
C---DOUBLE STRETCHING; INPUT MINIMUM SPACING------------------------(7)
C
  170       ARGI = STRETCH(INDEX)*REAL(2*NODE - NMBRNDS(INDEX) - 1)
            EXPI = EXP(ARGI)
           EXPII = 1.0/EXPI
           TANHI = (EXPI - EXPII)/(EXPI + EXPII)
C
            ARGN = STRETCH(INDEX)*REAL(NMBRNDS(INDEX) - 1)
            EXPN = EXP(ARGN)
           EXPNI = 1.0/EXPN
           TANHN = (EXPN - EXPNI)/(EXPN + EXPNI)
C
      ETA(INDEX) = 0.5*(1.0 + TANHI/TANHN)
C
      GO TO 300
C
C---DECREASING SPACING; INPUT STRETCHING FACTOR---------------------(8)
C
  180       PIDN = PI/(STRETCH(INDEX)*NMBRNDS(INDEX))
C
      ETA(INDEX) = 1.0 - TAN(PIDN*(NMBRNDS(INDEX) - NODE))
     &                   /TAN(PIDN*(NMBRNDS(INDEX) - 1))
C
      GO TO 300
C
C---INCREASING SPACING; INPUT STRETCHING FACTOR---------------------(9)
C
  190       PIDN = PI/(STRETCH(INDEX)*NMBRNDS(INDEX))
C
      ETA(INDEX) = TAN(PIDN*(     NODE      - 1))
```

```
     &             /TAN(PIDN*(NMBRNDS(INDEX) - 1))
C
      GO TO 300
C
C---USER INPUT STRETCHING FUNCTION---------------------------------(10)
C
  200 CONTINUE
C
C---STORE SPACING--------------------------------------------------------
C
  300 ETAS(INDEX,NODE) = ETA(INDEX)
C
  310        ETA(INDEX) = ETAS(INDEX,NODE)
C
      RETURN
      END
C
C*************************************************************************
C*****HGM*****************************************************************
C*************************************************************************
C
      SUBROUTINE HOLES(ANGLE,POINT,TANGENT,ICLOCK,IDUCT)
C-------------------------------------------------------------------------
C     CALCULATE POINT AND TANGENT ON HOLE
C
C       ANGLE = ANGULAR LOCATION OF POINT
C       POINT = POSITION AND FLOW VECTOR
C     TANGENT = TANGENT
C      ICLOCK = DIRECTION OF TANGENT
C-------------------------------------------------------------------------
C
      COMMON /INITC/   PI,RADDEG
C
      DIMENSION POINT(6),TANGENT(3),US(3),BC(3),REF(3)
C
           ANG = ANGLE/RADDEG
C
          SANG = SIN(ANG)
          CANG = COS(ANG)
C
      IF(IDUCT.EQ.1) THEN
C-------------------------------------------------------------------------
C     UPPER DUCT
C-------------------------------------------------------------------------
C---UNIT VECTOR ALONG AXIS
C
          A1 =  0.25506
          A3 =  0.092215
          A5 =  0.96252
C
C---INTERMEDIATE VECTOR
C
          A2 =  2.07828*CANG - 0.050785*SANG
          A4 =         5.564 + 2.14084*SANG
          A6 = -0.55072*CANG - 0.19165*SANG
C
C---REFERENCE POINT ON AXIS
C
      REF(1) = 9.445
      REF(2) = 5.564
```

```
      REF(3) = 0.0
                        ELSE
C-------------------------------------------------------------------------
C     LOWER DUCT
C-------------------------------------------------------------------------
C---UNIT VECTOR ALONG AXIS
C
          A1 = 0.40139
          A3 = 0.0
          A5 = 0.91591
C
C---INTERMEDIATE VECTOR
C
          A2 = 1.96921*CANG - 2.11734
          A4 =              2.150*SANG
          A6 = 0.37334 - 0.86299*CANG
C
C---REFERENCE POINT ON AXIS
C
      REF(1) = 7.32766
      REF(2) = 0.0
      REF(3) = 0.37334
C
                        END IF
C-------------------------------------------------------------------------
C     POSITION
C-------------------------------------------------------------------------
          RAD =  7.890
C
C---QUADRATIC EQUATION
C
            AA = A1*A1 + A3*A3 + A5*A5
C
            BB = 2.0*(A1*A2 + A3*A4 + A5*A6)
C
            CC = A2*A2 + A4*A4 + A6*A6 - RAD**2
C
           ARG = BB*BB - 4.0*AA*CC
C
C---DISTANCE ALONG AXIS
C
             C = 0.5*(-BB + SQRT(ARG))/AA
C
C---VECTOR FROM SPHERE ORIGIN TO POINT ON HOLE
C
         US(1) = C*A1 + A2
         US(2) = C*A3 + A4
         US(3) = C*A5 + A6
C
C---VECTOR FROM ORIGIN TO POINT ON HOLE
C
      POINT(1) = US(1) + 9.445
      POINT(2) = US(2)
      POINT(3) = US(3)
C-------------------------------------------------------------------------
C     TANGENT
C-------------------------------------------------------------------------
C---VECTOR FROM AXIS TO POINT ON HOLE
C
         BC(1) = POINT(1) - C*A1 - REF(1)
```

```
         BC(2) = POINT(2) - C*A3 - REF(2)
         BC(3) = POINT(3) - C*A5 - REF(3)
C
C---TANGENT AND FLOW VECTOR
C
      IF(ICLOCK.EQ.0) THEN
C
                      CALL CROSS(US,BC,TANGENT,60)
C
                      CALL CROSS(US,TANGENT,POINT(4),61)
C
                      ELSE
C
                      CALL CROSS(BC,US,TANGENT,62)
C
                      CALL CROSS(TANGENT,US,POINT(4),63)
C
                      END IF
      RETURN
      END
C
C*************************************************************************
C*****OUTPUT*************************************************************
C*************************************************************************
C
      SUBROUTINE OUTPUT(NUNIT,NODSTOR)
C-------------------------------------------------------------------
C   PRINTOUT AND STORE DATA
C-------------------------------------------------------------------
C               DOUBLE PRECISION NODE
                          REAL NODE
C
      COMMON /COUNTER/ NODESAV,NODETOT,NBNODES,NPLANE
      COMMON /HEADER/  ITITLE(20),LINE
      COMMON /INITA/   IDIM,MAPTEN,INCHES
      COMMON /IWRITE/  IWRTC,IWRTI,IWRTN
      COMMON /OUT/     NODE(5,10000)
      COMMON /UNITS/   NU5,NU6,NU19,NU20,NU21
      COMMON /ZONING/  ISECT,NSECT,IZONE,NMBRNDS(3)
C
C---TOTAL NUMBER OF PLANES
C
           NPLANE = NPLANE + 1
C
C-------------------------------------------------------------------
C---PRINT OUTPUT
C-------------------------------------------------------------------
C---INTERMEDIATE PRINT
C
  100 IF(IWRTI.GT.0 .AND. LINE.LE.56) THEN
C
                                      WRITE(NU6,1030)
C
                                      LINE = LINE + 3
C
                                      END IF
C
C---PRINT NODAL INFORMATION
C
      IPRINT = 1
```

```
C
      DO 120 I=1,NODSTOR
C
      IF(IWRTN.EQ.0 .OR. I.LT.IPRINT) GO TO 120
C
  110 IPRINT = I + IWRTN
      NODNUM = I + NODETOT
C
         LINE = LINE + 1
C
      IF(LINE.GE.60) THEN
C
                     WRITE(NU6,1000) ITITLE,ISECT,NSECT,IZONE
                     WRITE(NU6,1030)
C
                     LINE = 5
C
                     END IF
C
C
      WRITE(NU6,1040) NODNUM,(NODE(J,I),J=1,5)
C
  120 CONTINUE
C-----------------------------------------------------------------------
C   STORE OUTPUT
C-----------------------------------------------------------------------
C
  200 CALL BLKOUT(NUNIT,NODSTOR)
C
C---NODE COUNTERS
C
           NODETOT = NODSTOR + NODETOT
           NODESAV = NODESAV - NODSTOR
C
C---PRINT TOTAL NUMBER OF POINTS STORED
C
      IF(IWRTI.EQ.0) RETURN
C
                         LINE = LINE + 1
C
C
                    WRITE(NU6,1060) NODSTOR,NPLANE,NUNIT,NODETOT
C
      RETURN
C
C---FORMAT STATEMENTS
C
 1000 FORMAT(1H1,10X,20A4,13X,8H SECTION,I2,3H OF,I3,9H FOR ZONE,I3)
 1020 FORMAT(1X,I6,2(1X,F13.7),5X,F7.2,6X,F7.2,5X,I2)
 1030 FORMAT( / 44H   NODE       X              Y              Z   )
 1040 FORMAT(1X,I6,3(1X,F13.7),3X,2(2X,F7.2),3X,2(2X,F7.2),5X,I2)
 1050 FORMAT( 5X,I5,18H POINTS FROM PLANE,I4,15H STORED ON UNIT,I3,
     &              23H: TOTAL POINTS STORED =,I6 )
 1060 FORMAT(10X,I5,18H POINTS FROM PLANE,I4,15H STORED ON UNIT,I3,
     &              23H: TOTAL POINTS STORED =,I6 )
C
      END
C
C**************************************************************************
C*****OUTPUT*****************************************************************
```

```
C*********************************************************************
C
      SUBROUTINE PICTURE(IDRAW)
C---------------------------------------------------------------------
C   THIS ROUTINE DESCRIBES THE NOMENCLATURE
C---------------------------------------------------------------------
C
      COMMON /UNITS/   NU5,NU6,NU19,NU20,NU21
C
      WRITE(NU6,300)
      WRITE(NU6,310)
      WRITE(NU6,320)
      WRITE(NU6,330)
C
      RETURN
C
C---FORMAT STATEMENTS
C
C
  300 FORMAT( 40X,37H                      3D NOMENCLATURE
     1       /40X,19H                    IE
     2       /40X,19H                    IT
     3       /40X,34H                    IA       SURFACE 4
     4       /40X,32H                    I2         (TOP)
     5       /40X,50H          POINT 4   I                          POINT 3
     6       /40X,42H                    O------------------------O
     7       /40X,42H                  8/I         EDGE 3        7/I
     8       /40X,50H                 E/ I                      E/ I        S)
  310 FORMAT( 40X,50H                G/   I      SURFACE 1     G/   I        U
     1       /40X,50H               D/    I       (BACK)      D/    I    E   R
     2       /40X,50H              E/     EI                  E/     I    D   F
     3       /40X,50H POINT 8   O------DI-----------------O POINT 7 G    A
     4       /40X,50H           I       GI    EDGE 11      I      I    E   C
     5       /40X,50H   S       I       EI                 IE     I        E
     6       /40X,46H   U   E   I        I                 ID     I    2
     7       /40X,50H   R   D   I       4I     SURFACE 3   IG     I        6
     8       /40X,42H   F   G   I        I      (FRONT)    IE     I
     9       /40X,50H   A   E   I        O_________________I_____O____ETA1)
  320 FORMAT( 40X,51H   C     POINT 1 /5       EDGE 1      I1     /  POINT 2
     1       /40X,42H   E   1   I     /E                   I0    /6
     2       /40X,41H       2   I    /G                    I    /E
     3       /40X,40H   5       I   /D                     I   /G
     4       /40X,39H           I  /E                      I  /D
     5       /40X,38H           O/_________________________O/E
     6       /40X,44H   POINT 5          EDGE 9             POINT 6
     7       /40X,10H           /
     8       /40X,26H         /3        SURFACE 2
     9       /40X,25H       /A          (BOTTOM)                          )
  330 FORMAT( 40X, 8H     /T
     &       /40X, 7H    /E                                              )
C
C
      END
C
C*********************************************************************
C*****UTILITY********************************************************
C*********************************************************************
C
      SUBROUTINE VADD(CA,A,CB,B,C,UC)
C---------------------------------------------------------------------
```

```
C   VADD COMPUTES C,THE SUM OF VECTORS CA*A AND CB*B,WHERE CA AND
C   CB ARE SCALARS. UC IS A UNIT VECTOR DIRECTED ALONG C.
C-----------------------------------------------------------------------
C
      DIMENSION A(3),B(3),C(3),UC(3)
C
      SUM = 0.0
C
      DO 10 I=1,3
      C(I) = CA*A(I) + CB*B(I)
   10  SUM = SUM + C(I)*C(I)
C
      CMAG = SQRT(SUM)
C
                        RMAG = 0.0
      IF(CMAG.GT.0.0) RMAG = 1.0/CMAG
C
      UC(1) = C(1)*RMAG
      UC(2) = C(2)*RMAG
      UC(3) = C(3)*RMAG
C
      RETURN
      END
C
C***********************************************************************
C*****UTILITY***********************************************************
C***********************************************************************
C
      SUBROUTINE VDOT(A,B,C)
C-----------------------------------------------------------------------
C   VDOT COMPUTES C,THE DOT PRODUCT OF VECTORS A AND B.
C-----------------------------------------------------------------------
C
      DIMENSION A(3),B(3)
C
      C = 0.0
C
      DO 10 I=1,3
   10 C = C + A(I)*B(I)
C
      RETURN
      END
C
C
C***********************************************************************
C*****UTILITY***********************************************************
C***********************************************************************
C
      SUBROUTINE VMAG(VECTOR,VECMAG)
C-----------------------------------------------------------------------
C   VMAG DETERMINES THE MAGNITUDE OF A VECTOR
C-----------------------------------------------------------------------
C
      DIMENSION VECTOR(3)
C
      VECMAG = SQRT(VECTOR(1)**2 + VECTOR(2)**2 + VECTOR(3)**2)
C
      IF(VECMAG.LT.0.0000001) VECMAG = 0.0
C
      RETURN
```

```
      END
C
C*************************************************************************
C*****HGM*****************************************************************
C*************************************************************************
C
      SUBROUTINE FAIRING(PT1,PT2,POINT,TANGENT,SNORMAL)
C-------------------------------------------------------------------------
C   TRANSITION FROM BOWL TO DUCT
C
C       PT1 = POINT ON HOLE (ON SPHERE)
C       PT2 = POINT ON DUCT (END OF FAIRING)
C     POINT = POSITION
C   TANGENT = TANGENT
C   SNORMAL = SURFACE NORMAL
C-------------------------------------------------------------------------
C
      COMMON /COUNTER/ NODESAV,NODETOT,NBNODES,NPLANE
      COMMON /INITC/   PI,RADDEG
      COMMON /SPACING/ ISTRTCH(3),STRETCH(3),ETAS(3,200),ETA(3),DETA(3)
      COMMON /ZONING/  ISECT,NSECT,IZONE,NMBRNDS(3)
C
      DIMENSION PT1(3),PT2(3),POINT(6),TANGENT(3),SNORMAL(3)
      DIMENSION PTW(3),UC(3),CP(3),REF(3),VDUM(3)
      DIMENSION R1(3),R2(3),UN(3),UP(3)
C
      NODNUM = NODESAV + NODETOT + 1
C
C---VECTOR FROM SPHERE ORIGIN TO EDGE OF HOLE
C
        P1 = PT1(1) - 9.445
        P2 = PT1(2)
        P3 = PT1(3)
C
      PMAG = SQRT(P1*P1 + P2*P2 + P3*P3)
C
C---NORMALIZED
C
        S1 = P1/PMAG
        S2 = P2/PMAG
        S3 = P3/PMAG
C
                IDUCT = IZONE-3
C
C      WRITE(6,3030) IAXIS,JAXIS,KAXIS
C 3030 FORMAT(' IAXIS(2),JAXIS(3),KAXIS(1) =',3I10)
C
      IF(IDUCT.EQ.1) THEN
C-------------------------------------------------------------------------
C     UPPER DUCT
C-------------------------------------------------------------------------
C---AXIS
C
          U1 = 0.25506
          U2 = 0.092215
          U3 = 0.96252
C
C---INTERMEDIATE VALUES
C
        CADD = 5.564*U2
```

```
C
          A1 = S1*(1.0 - U1**2) - S2*U1*U2 - S3*U1*U3
          A2 = S2*(1.0 - U2**2) - S1*U1*U2 - S3*U2*U3
          A3 = S3*(1.0 - U3**2) - S1*U1*U3 - S2*U2*U3
C
          B1 = 5.564*U1*U2
          B2 = 5.564*(U2*U2 - 1.0)
          B3 = 5.564*U2*U3
C
C---REFERENCE POINT ON AXIS
C
      REF(1) = 9.445
      REF(2) = 5.564
      REF(3) = 0.0
C
C      WRITE(6,3010)
C 3010 FORMAT(' UPPER DUCT')
C
                          ELSE
C-----------------------------------------------------------------------
C     LOWER DUCT
C-----------------------------------------------------------------------
C---AXIS
C
          U1 = 0.40139
          U2 = 0.0
          U3 = 0.91591
C
C---INTERMEDIATE VALUES
C
        CADD = -2.11734*U1 + 0.37334*U3
C
          A1 = S1*(1.0 - U1*U1) - S3*U1*U3
          A2 = S2
          A3 = S3*(1.0 - U3*U3) - S1*U1*U3
C
          B1 = -2.11734*(U1*U1 - 1.0) + 0.37334*U1*U3
          B2 = 0.0
          B3 = -2.11734*U1*U3 + 0.37334*(U3*U3 - 1.0)
C
C---REFERENCE POINT ON AXIS
C
      REF(1) = 7.32766
      REF(2) = 0.0
      REF(3) = 0.37334
C
C      WRITE(6,3020)
C 3020 FORMAT(' LOWER DUCT')
C
                          END IF
C-----------------------------------------------------------------------
C     CALCULATE END POINT OF FAIRING
C-----------------------------------------------------------------------
C---QUADRATIC EQUATION
C
          AA = A1*A1 + A2*A2 + A3*A3 - 1.0
C
          BB = 2.0*(A1*B1 + A2*B2 + A3*B3 + 5.930)
C
          CC = B1*B1 + B2*B2 + B3*B3 - 35.1649
```

```
C
          DD = BB*BB - 4.0*AA*CC
C
         SDD = SQRT(DD)
C
        RHO1 = 0.5*(-BB + SDD)/AA
        RHO2 = 0.5*(-BB - SDD)/AA
C
C---RADIUS FROM SPHERE ORIGIN TO ARC CENTER
C
                        RHO = RHO1
      IF(RHO2.GT.RHO1) RHO = RHO2
C
C---ARC RADIUS
C
         RAD = RHO - 7.890
C
C---RADIUS FROM DUCT CENTERLINE TO ARC CENTER
C
          CN = 1.96 + RAD
C
C---UNIT VECTOR FROM DUCT CENTERLINE TO ARC CENTER
C
          C1 = (RHO*A1 + B1)/CN
          C2 = (RHO*A2 + B2)/CN
          C3 = (RHO*A3 + B3)/CN
C
C---SWEEP ANGLE OF ARC
C
          SN = C1*S1 + C2*S2 + C3*S3
C
         ANG = ACOS(SN)
C
C---VECTOR FROM ORIGIN TO ARC CENTER
C
         DC1 = RHO*S1 + 9.445
         DC2 = RHO*S2
         DC3 = RHO*S3
C
C---DISTANCE ALONG DUCT CENTERLINE
C
          CA = RHO*(S1*U1 + S2*U2 + S3*U3) - CADD
C
C---VECTOR FROM ORIGIN TO END POINT OF FAIRING
C
      PTW(1) = CA*U1 + 1.96*C1 + REF(1)
      PTW(2) = CA*U2 + 1.96*C2 + REF(2)
      PTW(3) = CA*U3 + 1.96*C3 + REF(3)
C
C                                      ANGL = ANG*RADDEG
C      WRITE(6,3000) NODNUM,RHO,RAD,CA,ANGL,C1,C2,C3,(PTW(I),I=1,3)
C 3000 FORMAT(' NODE =',I10 / ' RHO,RAD,CA,ANGL =',4(2X,F12.5)
C     1                     / '         C1,C2,C3 =',3(2X,F12.5)
C     2                     / '              PTW =',3(2X,F12.5))
C
      JUNCTION = 7
C
      IF(IAXIS.LT.JUNCTION) THEN
C-------------------------------------------------------------------------
C     FAIRING SURFACE
```

```
C-----------------------------------------------------------------------
C---VECTOR FROM ARC CENTER TO HOLE
C
       R1(1) = PT1(1) - DC1
       R1(2) = PT1(2) - DC2
       R1(3) = PT1(3) - DC3
C
C---VECTOR FROM ARC CENTER TO END OF FAIRING
C
       R2(1) = PTW(1) - DC1
       R2(2) = PTW(2) - DC2
       R2(3) = PTW(3) - DC3
C
C---LOCAL COORDINATE SYSTEM
C
      CALL CROSS(R1,R2,UN,70)
C
      CALL CROSS(R1,UN,UP,71)
C
      CALL CROSS(UN,UP,R1,72)
C
      CALL VDOT(UP,R2,UPE)
C
      IF(UPE.LT.0.0) CALL CROSS(UN,R1,UP,73)
C
C---ARC ANGLE ALONG FAIRING
C
          THET = ANG*(FLOAT(IAXIS) - 1.0)/3.0
C
          CANG = COS(THET)
          SANG = SIN(THET)
C
C---UNIT VECTOR FROM ARC CENTER TO POINT ON FAIRING
C
      CALL VADD(CANG,R1,SANG,UP,VDUM,UC)
C
C---POSITION
C
        POINT(1) = DC1 + RAD*UC(1)
        POINT(2) = DC2 + RAD*UC(2)
        POINT(3) = DC3 + RAD*UC(3)
C
C        WRITE(6,3050) PT1,PT2,RHO1,RHO2,POINT
C 3050 FORMAT('          PT1,PT2 =',6(2X,F12.5)
C     1     / ' RHO1,RHO2,POINT =',14X,5(2X,F12.5))
C
C---TANGENT
C
      CALL VADD(CANG,UP,-SANG,R1,VDUM,TANGENT)
C
      CE = TANGENT(1)*U1 + TANGENT(2)*U2 + TANGENT(3)*U3
C
      IF(CE.LT.0.0) THEN
C
                    TANGENT(1) = -TANGENT(1)
                    TANGENT(2) = -TANGENT(2)
                    TANGENT(3) = -TANGENT(3)
C
                    END IF
C
```

```
C---NORMAL
C
      SNORMAL(1) = -UC(1)
      SNORMAL(2) = -UC(2)
      SNORMAL(3) = -UC(3)
C
                      ELSE
C-----------------------------------------------------------------------
C     DUCT SURFACE
C-----------------------------------------------------------------------
C---RATIO BETWEEN END OF FAIRING AND END OF SECTION
C
C     DUCT SURFACE
C-----------------------------------------------------------------------
C---RATIO BETWEEN END OF FAIRING AND END OF SECTION
C
      TOTND = FLOAT(NMBRNDS(2))
      RATIO = (TOTND - IAXIS) / (TOTND - JUNCION)
C
      STR21 = 3.0
C
        X1 = RATIO*STR21/2.0
       ETA1 = (EXP(X1) - EXP(-X1))/(EXP(X1) + EXP(-X1))
C
        X2 = STR21/2.0
       ETA2 = (EXP(X2) - EXP(-X2))/(EXP(X2) + EXP(-X2))
C
        EPS = 1.0 - ETA1/ETA2
C
       EPS1 = 1.0 - EPS
C
C---VECTOR FROM ORIGIN TO POINT ON DUCT
C
      CALL VADD(EPS1,PTW,EPS,PT2,CP,VDUM)
C
C---POSITION
C
        POINT(1) = CP(1)
        POINT(2) = CP(2)
        POINT(3) = CP(3)
C
C---TANGENT
C
      TANGENT(1) = U1
      TANGENT(2) = U2
      TANGENT(3) = U3
C
C---NORMAL
C
      SNORMAL(1) = C1
      SNORMAL(2) = C2
      SNORMAL(3) = C3
C
                      END IF
C-----------------------------------------------------------------------
C     FLOW VECTOR
C-----------------------------------------------------------------------
        POINT(4) = TANGENT(1)
        POINT(5) = TANGENT(2)
        POINT(6) = TANGENT(3)
```

```
C
      RETURN
      END
C
C**EOR**
C**EOI**
```